Grado en Matemática Computacional

Estancia en Prácticas y Proyecto Final de Grado

Estudio de las ecuaciones de Shallow Water. Resolución numérica en una dimensión

Autor:
Rubén Ferrandis Naixes

Supervisor:
Héctor Llop Plà
Tutor académico:
Vicente Martínez García

Fecha de lectura: __ de _____ de 2020
Curso académico 2019/2020

Resumen

En este documento se tratan tanto el trabajo final en el grado de Matemática Computacional, dirigido hacia el estudio del comportamiento de un fluido en una dimensión, así como también una parte dedicada a la estancia en prácticas en la empresa Pixelcom Ingenieria S.L.

La primera parte del documento se centra en la estancia en las prácticas extracurriculares, donde aprendí a programar en WPF para el desarrollo de software en aplicaciones de escritorio. Posteriormente pasé a un plano más enfocado al desarrollo web con React JS, a lo que más tarde se le sumó el desarrollo de aplicaciones móviles para las plataformas con sistema Android y iOS, utilizando el lenguaje React-Native.

La segunda parte del documento está enfocada a una parte más teórica, la cual está dedicada al estudio y deducción de modelos de aguas someras en una dimensión mediante el uso de modelos clásicos, que describen de forma aproximada el flujo de distintos escenarios donde se produce el fenómeno conocido como Shallow Water.

Palabras clave

Aguas someras, superficie libre, leyes de conservación, Saint-Venant, Lax-Friedrichs.

Abstract

In this document it is treated the final project of the Computational Mathematics degree, directed toward the work of the research of the behaviour of a fluid in one dimension, as well as a part of the internship in Pixelcom Ingenieria S.L. company.

The first section of the document is focused on the stay in the extracurricular practices, where I learned to program on WPF for the software development of desktop applications. Then, I moved to a more focused web development plane with React JS, to which was later added, the mobile applications development for the platforms with Android and iOS systems, using the React-Native language.

The second section of the document is focused on a more theoretical part, which is dedicated to the study and deduction of shallow water models in one dimension through the use of classic models, that roughly describe the flow of different scenarios where the phenomenon known more commonly as Shallow Water occurs.

Keywords

Shallow water, free surface, conservation laws, river flows, Saint-Venant.

Índice general

Capítulo 1

Introducción

1.1. Contexto y motivación del proyecto

Este documento tiene como principal finalidad la del desarrollo del proyecto relacionado con el Trabajo de Fin de Grado, que trata sobre el desarrollo teórico del comportamiento de fluidos en aguas poco profundas, más conocido como aguas someras o Shallow Water. Además, hay un breve tema relacionado con la estancia en prácticas, que difiere bastante del trabajo de investigación teórico, ya que son unas prácticas enfocadas al desarrollo de software.

Elegí este proyecto por dos principales razones. La primera es que el profesor a cargo del proyecto es Vicente Martínez, profesor y tutor con el cual tuve la suerte de compartir un curso en la asignatura de Cálculo numérico avanzado el año anterior, y quedé bastante satisfecho con su didáctica durante ese curso, por lo que sé que sería de gran ayuda durante el actual curso con el tema de la modelización, debido a sus años de experiencia en este campo. La segunda razón fue que, de primeras, el trabajo que me propuso ya me llamó mucho la atención, y me surgió esa curiosidad por aprender más a fondo sobre temas tan cercanos al mundo real, como lo son la formación de olas, tsunamis u otros fenómenos físicos que ocurren día a día, de los que sabía más bien poco y tenía por seguro que había una gran parte matemática detrás que indagara en estas formaciones.

Además de la parte teórica, decidimos incluir una parte práctica final donde pudiéramos aplicar las ecuaciones a las que finalmente llegaríamos, a través del software Matlab, para que el trabajo pudiera quedar de una forma aún más completa.

El trabajo está distribuido en 2 partes. Tras esta breve introducción empieza la primera parte, dedicada a mi estancia en prácticas en Pixelcom Ingeniería S.L., donde hablo de mi

experiencia en la empresa, aplicando los conocimientos de programación adquiridos durante los estudios de grado. La segunda parte, tal y como he mencionado, se trata de un desarrollo teórico a partir de dos leyes de la conservación, para llegar a las ecuaciones de Shallow Water, a partir de las cuales, haremos una demostración práctica de cómo se modelizan dichas ecuaciones para solventar un problema (entre otros) que más adelante describiremos.

Capítulo 2

Estancia en prácticas

2.1. Introducción

En este capítulo se va a detallar la estancia en prácticas realizadas durante el curso en la empresa Pixelcom Ingenieria S.L., en Alquerías del Niño Perdido, durante los meses comprendidos entre finales de Julio y Diciembre, y cuyo proyecto fue supervisado por Héctor Llop Plà. La empresa, especializada en hardware de visualización y de control en el cual lleva años destinados, sobre todo dirigido hacia el mundo del motorsport, empieza desde hace unos años a enfocarse también en el desarrollo de software para dicho campo.

En esta parte del software, que es donde se enfoca la estancia en prácticas, se desarrollan soluciones timing para la gestión y control de cualquier pista de karting. Durante la estancia, cuento con la supervisión del responsable, Héctor Llop Plà, gerente del departamento de informática y de la empresa junto a su socio José Manuel.

2.2. Objetivos del proyecto formativo

Los objetivos del proyecto, los cuales parten de varias formaciones, podemos separarlos en varios procesos distintos, siendo simultáneos dos de ellos durante toda mi estancia.

El primer proceso se trata en aprender y poner en práctica el lenguaje WPF, del cual hablaré más al detalle en adelante, me permite dominar gran parte del software de la empresa con el que trabajan nuestros clientes. Empiezo con una formación, la cual voy complementando posteriormente gracias a la resolución de tareas reales ya con estos programas.

El segundo proceso también empieza con otra formación, pero con un enfoque dirigido hacia el desarrollo de aplicaciones web. Al igual que la anterior formación, empiezo a aprender React, el cual me toma aproximadamente algo más de una semana para asentar las bases, y a partir de aquí, empiezo a programar varias tareas de dificultad reducida en algunos proyectos web de la empresa.

Los anteriores 2 objetivos tenían como finalidad aprender los dos lenguajes en los que desarrollamos aplicaciones, pero el tercero no es un objetivo directo sobre programación, sino que, se trata del Scrum, una metodología de trabajo que me permitió aprender a trabajar en equipo de forma distinta y que más adelante detallaré.

2.3. Breve Resumen del proyecto realizado

En este apartado voy a tratar de resumir los 6 meses de mi estancia en prácticas entrando solamente un poco en detalle en lo referente al proyecto de formación de la empresa.

Los primeros días en la empresa, mi tarea no fue nada más que empezar a familiarizarme con mi entorno de trabajo, por lo tanto, se me proporcionó un entorno junto con el equipo de desarrollo, además de mi nueva herramienta de trabajo. Durante estos primeros días instalé todos los programas necesarios que me permitirían desarrollar durante la estancia y, también, todos los programas de escritorio que utilizaban los clientes para así familiarizarme con estos.

Tras tener el entorno de desarrollo bien estructurado, empecé lo que podría llamarse como la primera fase, ya que es donde empecé realmente la formación. En esta fase, y haciendo uso de un nuevo IDE con el que nunca había trabajado (Visual Studio), empecé mi formación para el desarrollo en la tecnología Windows Presentation Foundation, o más conocido comúnmente como WPF.

Esta tecnología permite crear aplicaciones de escritorio con el uso tanto de la lógica como de la interfaz gráfica de forma simultánea. Durante la formación aprendí a programar en los lenguajes de los que se componía esta tecnología, y que más adelante detallaré. Además, aprendí a programar usando el MVVM (Model, View, View-Model), el cual se puede resumir en un patrón de programación donde conseguimos separar cualquier código en 3 partes bien diferenciadas: La lógica (Model), la Vista de la aplicación o Interfaz (View) y el intermediario o puente que hace de conexión entre estas dos partes (View-Model).

Tras terminar mi formación al cabo unos días, se me empezaron a proporcionar varios bugs de menor dificultad que tenían algunos de estos programas para solucionarlos. Esto me permitía, a su vez, empezar a familiarizarme con el código de estos programas el cual era inmenso al lado de los de la formación, pero con una misma estructura.

A partir de aquí, podríamos decir que empieza una segunda fase de la formación. Mientras iba resolviendo estos bugs de menor importancia empecé con la segunda formación del proyecto, la cual estaba enfocada al desarrollo web. El lenguaje con el que desarrollé para web se trata de React, una biblioteca JavaScript diseñada para crear interfaces de usuario.

Al igual que la anterior formación, empecé con unos días de formación básica y a partir de aquí, empecé a desarrollar en alguno de los programas que tenía la empresa, ya fuera resolviendo pequeños bugs, o implementando cosas nuevas de baja dificultad. A medida que iba programando y desarrollando nuevos conocimientos, se me empezaban a proporcionar también tareas de mayor dificultad, hasta un punto en donde empecé ya, más seriamente, a añadir nuevos módulos o "features", como se las conoce en la jerga de la informática, a las aplicaciones de las que disponía la empresa.

A parte de aprender nuevos lenguajes de programación, con mucha más presión y nivel de lo que se suele exigir durante los estudios de grado, aprendí a trabajar con la metodología de Scrum. Esta metodología, se trata de un conjunto de buenas prácticas en forma de colaboración con todo el equipo, con el fin de obtener el mayor rendimiento y resultados posibles en el proyecto.

Me costó adaptarme a esta forma de trabajo, ya que era algo completamente nuevo para mi. Creo sinceramente que merece la pena el esfuerzo, porque una vez asientas las bases, el trabajo se vuelve mucho más limpio y fluido de lo que se haría de otra forma. Además, hay algún software especializado sobretodo para el desarrollo de proyectos, que facilita mucho el trabajo con esta metodología.

Finalmente, también aprendí una nueva tecnología llamada React-Native, la cual permite el desarrollo de aplicaciones móviles tanto para Android como para iOS de forma nativa con lenguaje JavaScript. Es un potente framework de desarrollo móvil que traduce el código JavaScript a nativo en Android e iOS. Es posiblemente la tecnología a la que más me costó adaptarme, pero, a su vez, la que me pareció más interesante de todas.

2.4. Lenguajes de programación y entornos de desarrollo

Durante la estancia en prácticas he aprendido a programar y trabajar con software muy variado, el cual se describe en este apartado.

2.4.1. WPF

Gran parte del proyecto en mi estancia en prácticas se puede dividir principalmente en dos lenguajes de desarrollo, el primero fue WPF. Windows Presentation Foundation, o conocido de forma simplificada como WPF, es una tecnología de Microsoft que permite el desarrollo de aplicaciones de escritorio en Windows, donde se usa el lenguaje C# para la lógica y lenguaje XAML para las interfaces.

Esta tecnología requiere de dos lenguajes bastante distintos, el primero es XAML, el cual es un lenguaje basado en XML. Este lenguaje está relacionado con la parte de la interfaz gráfica de las aplicaciones, basado en etiquetas y atributos, y nos permite ir visualizando esta parte gráfica a medida que escribimos código. Es un lenguaje con una base muy fácil de entender incluso si nunca antes has trabajado con él (como en mi caso), pero se va complicando a medida que tienes que unir los componentes con su parte lógica.

```
    <Button
        Margin="20,0"
        HorizontalAlignment="Center"
        Click="Button_Click_SelectLogo">
        Load Logo
    </Button>
    <StackPanel HorizontalAlignment="Center" Orientation="Horizontal">
        <Slider
            Name="BrightnessSlider"
            Width="120"
            VerticalAlignment="Center"
            IsSnapToTickEnabled="True"
            Maximum="10"
            Minimum="0"
            TickFrequency="1" />
        <TextBlock
            Name="TextBoxBrightness"
            Width="20"
            VerticalAlignment="Center"
            FontSize="14"
            FontWeight="Bold"
            Foreground="White"
            IsEnabled="False"
            TextAlignment="Center" />
    </StackPanel>
</StackPanel>
```

Fig. 1: Ejemplo de código XAML.

El segundo lenguaje que compone WPF se trata de C#, lenguaje que nunca antes había visto tampoco, pero al ser un derivado de lenguaje C/C++ y tener un modelo de objetos similar al de Java, no supuso ningún problema a la hora de implementar nuevo código. Con C# creamos la parte lógica de nuestros programas, y que, se asemeja más a los típicos programas que hemos podido realizar durante los estudios de grado.

```
private void AsignarKart(KartData kartData)
{
    try
    {
        // Asignamos el kart al participante
        KartAssignment(kartData);

        // Indice del participante.
        var indiceParticipanteProximo = ListaParticipantes.IndexOf(ParticipanteSelected) + 1;

        //Seleccionamos automáticamente el siguiente participante
        if (indiceParticipanteProximo < ListaParticipantes.Count)
        {
            ParticipanteSelected = ListaParticipantes[indiceParticipanteProximo];
        }

        Common.RefreshICollectionView(ListaParticipantes);
        Common.RefreshICollectionView(ListaKarts);
    }
    catch (Exception ex)
    {
        App.AddLineInPosErrorsLog(ex, "Error Asign Kart");

        _parametrosGlobalesAction.ShowTextSpecialEventNotifyierCloseWindow("Error Asign Kart", 3, true,
            _parametrosGlobalesAction.GetWindowInicializada(Enumerados.TeVentanasInicializadasMain.MensajeWarningWindowView));
    }
}
```

Fig. 2: Ejemplo de código C#.

Finalmente, estas dos partes de lógica e interfaz, se unen a través del patrón MVVM(Model-View-View Model) para formar el programa. Este patrón sirve para separar las partes del proyecto en la Vista (parte gráfica), el Modelo (parte lógica) y la Vista-Modelo o Controlador, que es el encargado de unir estas dos partes y mantener su funcionamiento.

2.4.2. Visual Studio

Para completar el apartado anterior, describiremos brevemente lo que vendría a ser el entorno de desarrollo, el cual tiene la función de permitirnos escribir y compilar código. Visual Studio, es el entorno de desarrollo integrado (IDE), con el que se trabaja en WPF, aunque no limitado solo a este, ya que, también es compatible con otros lenguajes de programación entre los que destacan C++, Java, Python, Ruby y Php.

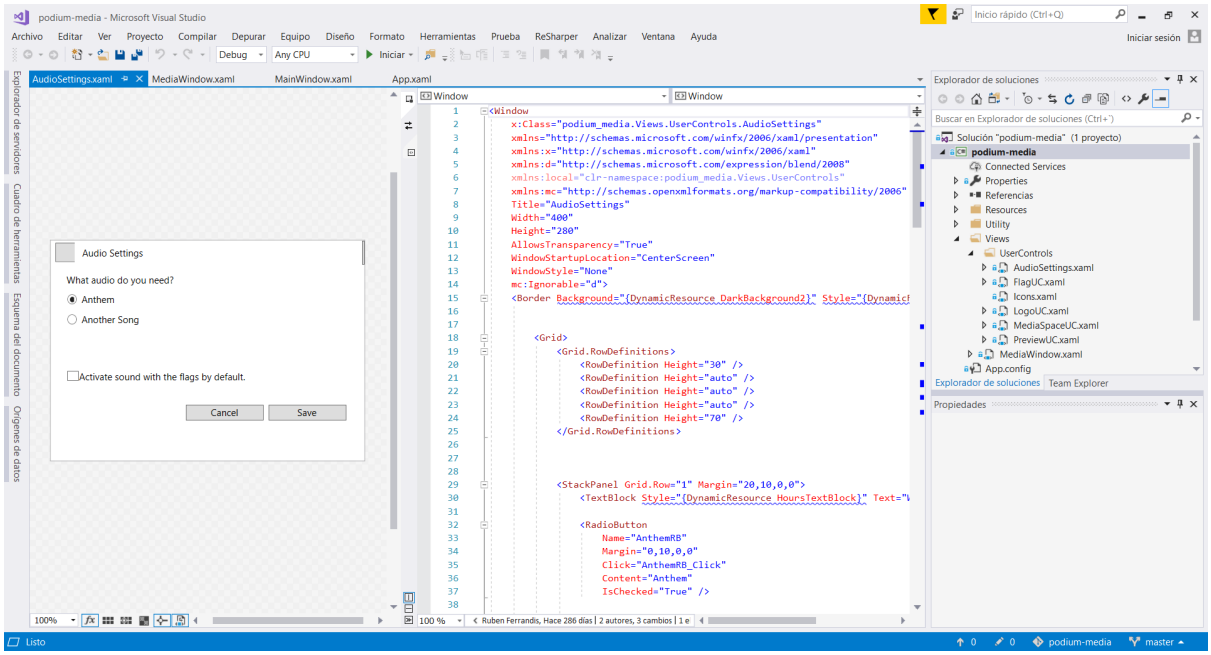


Fig. 3: Visual Studio (IDE).

2.4.3. React

Tal y como he mencionado en el primer apartado sobre las tecnologías empleadas, he implementado código durante la estancia en dos lenguajes principales, el primero ya descrito es WPF y el segundo se trata de React. React JS, se trata de una biblioteca JavaScript diseñada para crear interfaces de usuario, cuyo desarrollo está enfocado hacia desarrollo de aplicaciones en una sola página.

Esta biblioteca, al igual que el anterior lenguaje del que he hablado, usa un patrón de diseño MVVM o MVC y nace del problema de lentitud de la famosa aplicación Facebook. La diferencia respecto a otros lenguajes es que React usa un DOM (Document Object Model) [1] virtual propio en vez de usar el del navegador. Este DOM propio, al producirse un cambio del estado (State) de un componente, se encarga de ver las diferencias entre el DOM real y el DOM virtual, lo cual le permite aplicar cambios para el único o únicos componentes que se han modificado, en vez de realizar cambios para todo el DOM, con lo que esta operación de renderizado es mucho más eficiente.

[1] El DOM (Modelo de objetos del documento) se trata de una interfaz de plataforma, el cual permite añadir, cambiar o acceder de forma dinámica el contenido de documentos con lenguaje Javascript.

```
        <Button
          icon="add"
          intent="primary"
          text="Añadir"
          onClick={() => {onAdd(templates);}}/>

        <Button
          style={{ marginLeft: '5px' }}
          intent="success"
          icon="refresh"
          text="Refrescar"
          onClick={() => {onRetrieveCurrentTemplates(selectedCompany.id);}}/>
      </div>
    </div>
  );

  const filterTemplates = (templates, selectedCompany) => {
    const templatesFiltered = [];
    if (templates) {
      templates.forEach(template => {
        if (
          template.idCompany != null &&
          template.idCompany === selectedCompany.id
        ) {
          templatesFiltered.push(template);
        }
      });
      return templatesFiltered;
    }
    return [];
  };
```

Fig. 4: Ejemplo de código en React.

2.4.4. React-Native

React-Native es una biblioteca de JavaScript desarrollada también en Facebook e Instagram, utilizada para la implementación de aplicaciones móviles nativas en Android e iOS. Se programa en JavaScript y en JSX (para la creación de componentes), el cual tiene una sintaxis muy similar a HTML.

A diferencia del anterior mencionado React JS, el cual solo podría ejecutar aplicaciones nativas en un móvil dentro de un navegador, react-native si que puede ejecutar aplicaciones de forma nativa, lo cual tiene sus ventajas y sus desventajas. La principal ventaja sería el desarrollo para dos plataformas móviles en un lenguaje de programación sin necesidad de aprender nativo para Android o para iOS. Además, las apps nativas son propensas a tener una gran cantidad

de bugs. Por otra parte, una gran desventaja es el limitado acceso al hardware y a los APIs nativos como GPS o notificaciones push, los cuales son bastante más tediosos a la hora de implementarlos dentro de cualquier aplicación.

```
export const Users = ({ users, error }) => {
  const trans = useTrans()

  if (!isUndefined(error)) {
    return (
      <Text color="black" type="regular">
        {error}
      </Text>
    )
  }

  return (
    <View>
      <Text color="black" type="regular" style={styles.title}>
        {trans('users.title')}
      </Text>
      <FlatList
        data={users}
        renderItem={({ item: user }) => <UserItem user={user} />}
        keyExtractor={user => user.getId().toString()}
      />
    </View>
  )
}
```

Fig. 5: Ejemplo de código en React-native.

2.4.5. SQL

Poca presentación necesita SQL. El lenguaje SQL es un lenguaje de consultas que permite la gestión de información por medio de un sistema de gestión de bases de datos relacionales.

Es de los lenguajes más útiles y potentes a día de hoy, debido a su gran capacidad para manejar bases de datos relacionales con una inmensa cantidad de información. Durante los estudios de grado, hay una asignatura específica solo para este lenguaje donde prácticamente lo exprimimos haciendo ejercicios de consultas hasta la saciedad, que a priori parece bastante denso, pero al terminar el semestre tienes una muy buena base de conocimientos sobre la gestión de bases de datos.

Durante mi estancia en prácticas, uso consultas de muy bajo nivel, porque solamente la he

usado para ver información puntual en un determinado momento. A medida que pasaban las semanas empezaba a utilizarla más, debido a que mis tareas empezaban requerir tratamiento más continuo de la información almacenada, hasta un punto de necesitar comprobar información a diario. Aun así, las consultas de mayor dificultad que llegué a realizar durante mi estancia en prácticas, no requirieron más que alguna agrupación para la visualización de información muy general.

2.4.6. MySQL Workbench

MySQL Workbench es la herramienta que utilicé durante la estancia para la visualización de la información que había en las bases de datos. Este software es una herramienta visual de diseño de bases de datos que permite la gestión, administración, diseño y mantenimiento para el sistema de bases de datos MySQL.

Es una herramienta muy práctica que me permitía consultar la información que necesitaba en cada momento, sin la necesidad de escribir y reescribir las consultas una y otra vez, debido a que una función de gran utilidad es la de pre-cargar consultas, con lo que solo debía de buscar la tabla que necesitaba y ejecutarla.

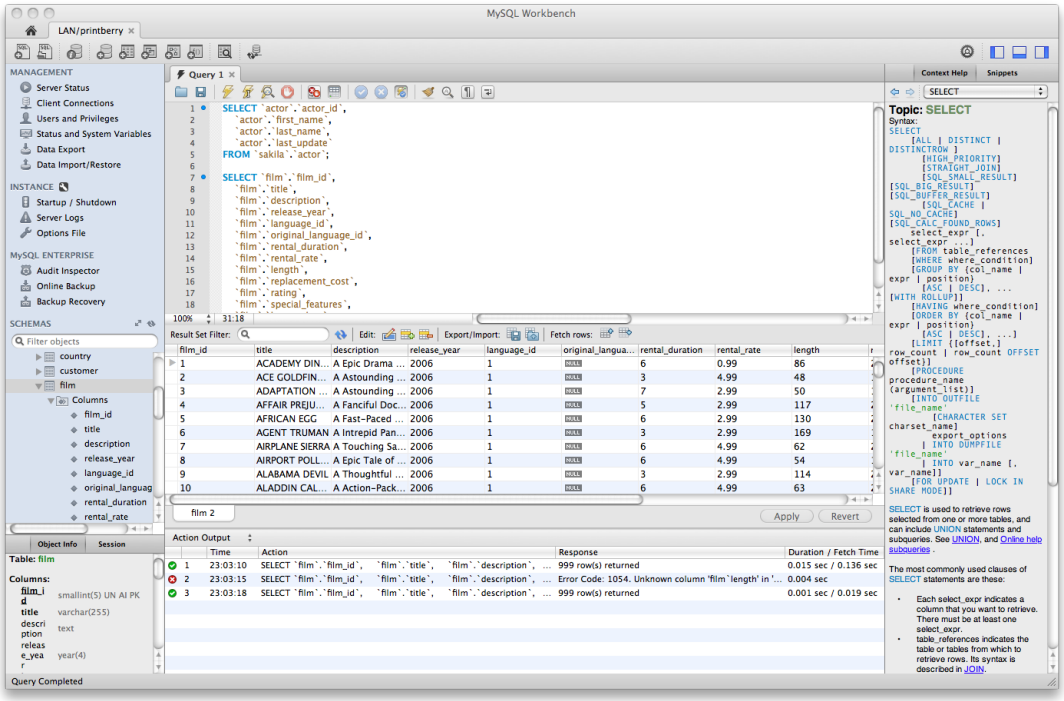


Fig. 6: MySQL Workbench.

2.4.7. Gitlab

La frase "Por último y no menos importante", podría aplicarse perfectamente como inicio para la descripción de este apartado, pero creo que le queda corto porque considero que este fue sin duda el software más importante con el que trabajé y del cual tenía total desconocimiento.

Gitlab es un servicio web que sirve principalmente como control de versiones y desarrollo de software colaborativo basado en Git, además, sirve como gestor de repositorios, y contempla almacenamiento para wikis y un sistema de seguimiento de errores. Cuando empecé mi estancia en prácticas no sabía siquiera lo que era el control de versiones o las ramas (*branches*), y a día de hoy, no me imagino programando ni realizando ningún proyecto sin dicha funcionalidad. A decir verdad, es la herramienta que más tiempo tardé en aprender porque no se aprendía de golpe y porrazo, sino que poco a poco y dependiendo de las tareas que se me proporcionaban, iba utilizando unas funcionalidades u otras.

Al permitir el almacenamiento de wikis, la empresa tenía sus propios manuales sobre diversos temas, como el control de versiones y las distintas ramas que se usan, el funcionamiento de la metodología Scrum, el software necesario a instalar para desarrollar ciertos proyectos, ciertas instrucciones para la escritura de código común en equipo, etc. Las wikis son el único apartado que utilicé las primeras semanas para la constante revisión de estos manuales sobre cosas que desconocía. Ya más adelante empecé a tratar con apartados más sofisticados como la gestión de la información sobre los proyectos, las distintas versiones entre los códigos, cómo sacar versión de un programa en producción o en integración, etc.

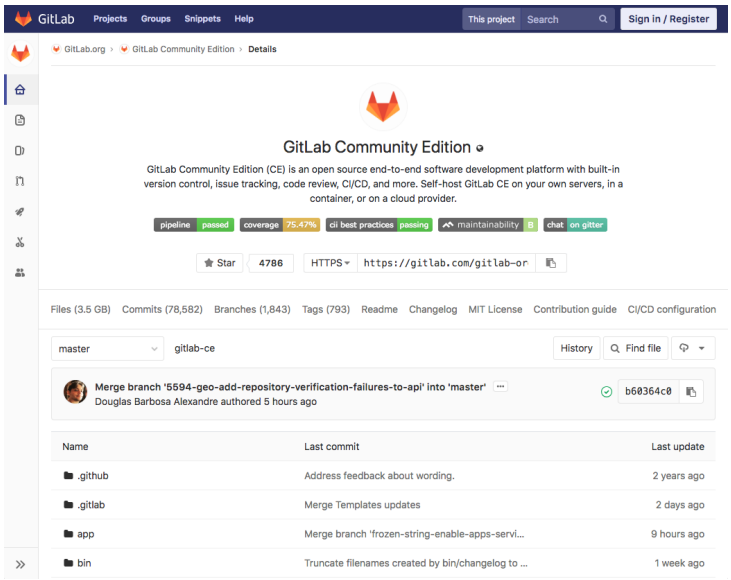


Fig. 7: Interfaz de Gitlab.

2.5. Explicación detallada el proyecto realizado en la empresa

Como he mencionado con anterioridad, el principal enfoque del proyecto estaba dirigido al aprendizaje de los lenguajes de programación que usábamos en la empresa y a su metodología de trabajo. Empecé el proyecto siguiendo una formación debido a mi desconocimiento por los lenguajes que tenía que dominar, para así, poco a poco, empezar a implementar nuevas funcionalidades o resolver bugs de los programas.

Voy a dividir mi estancia en prácticas en 5 apartados, porque creo que hay ciertas situaciones que me permiten marcar un antes y un después durante la estancia.

2.5.1. Iniciación en WPF

Los primeros dos días de mi estancia en Pixelcom, me proporcionaron todos los programas de los que disponía la empresa, junto con los respectivos manuales que se solían proporcionar a los clientes, los cuales me tuve que leer para entender todas las aplicaciones y testear todas las funcionalidades posibles. Esto me permitió de primera mano familiarizarme con los programas con los que iba a tener que trabajar.

A los dos días siguientes, que terminé de familiarizarme con los programas, empecé mi aprendizaje con el lenguaje de programación WPF (Windows Presentation Foundation), que unía las partes de lógica (usando C#) y diseño (usando XAML) de una forma muy dinámica y fácil de aprender para implementar aplicaciones de escritorio para el sistema operativo Windows. Durante los siguientes 10 días estuve aprendiendo dicho lenguaje, que finalmente aprendí gracias también en gran parte a mis compañeros de trabajo, los cuales resolvían todas mis dudas.

Además de aprender a programar en esta nueva tecnología, durante estas dos primeras semanas, empecé a adaptarme a la dinámica de la empresa, a aprender su filosofía y su forma de trabajar. En cuanto a las tareas, llamadas *issues*, una vez asignadas, las desfragmentamos todo lo posible para tenerlas de forma muy detallada, y así intentar tener claro desde el principio todas las posibles dudas que puedan surgir durante su resolución o implementación.

Una vez terminé la formación en WPF, empecé a tomar *issues* que eran en su mayoría pequeños bugs, ya que no estaba lo suficientemente preparado como para implementar nuevos módulos o nuevas *"features"*, como se les suelen llamar.

2.5.2. Iniciación en React

Tras pasar unos días resolviendo bugs de las aplicaciones Windows, tanto mi supervisor como mis compañeros decidieron que ya era hora de empezar con la formación dirigida al diseño web, que es lo que se pretendía desde un primer momento. Durante mis 4 años en la carrera no había visto nada de diseño web, por lo que era también nuevo para mí, aunque esto no supuso ningún problema finalmente a la hora de aprender.

Los primeros días iba un poco perdido, no entendía para que servían el estado o las propiedades que tan importantes eran en React, y, a decir verdad, al terminar la formación seguía sin tenerlo claro. Al terminar la formación seguí el mismo patrón de trabajo que con el anterior lenguaje de programación, es decir, se me proporcionaron varias issues de bajo nivel en React que me permitieran aplicar lo aprendido para así ir viendo como teníamos hechos los programas y que pudiera aprenderlo.

Estas issues, como digo, no tenían gran dificultad y por eso las he catalogado como de "bajo nivel". Por ejemplo, en la primera que realicé, simplemente se tenía que añadir un nuevo campo en el componente "*Card*", que permitiera visualizar el identificador de cada empresa. En la Figura 8 se puede ver lo simple que fue realizarlo, ya que, solamente tenía que añadir un nuevo componente tal y como el que había dentro de la tarjeta y añadir el campo que quería mostrar.

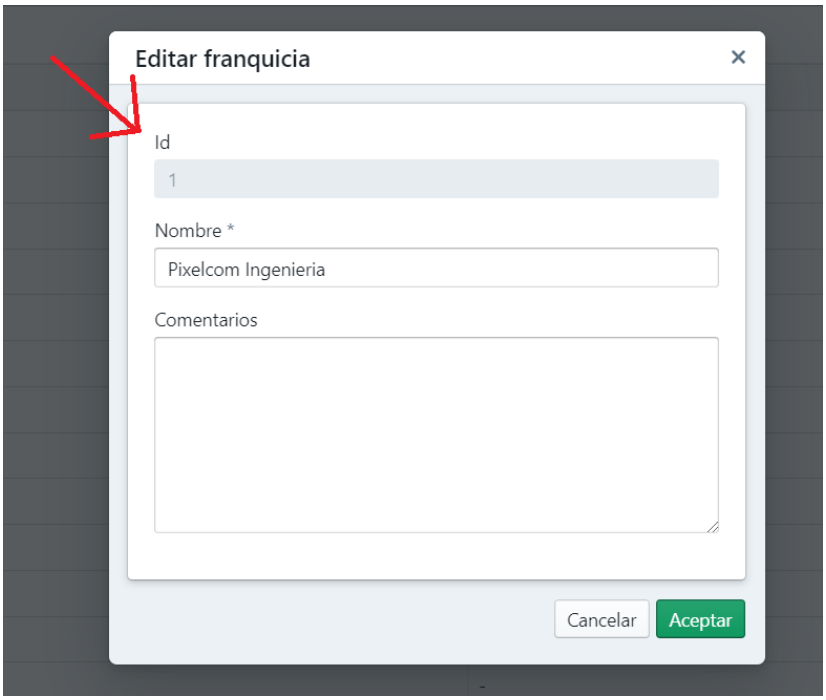


Fig. 8: Tarjeta de información React con id.

Para mí la dificultad radicaba en pasar de las aplicaciones de la formación, compuestas por un par de ficheros, a una aplicación entera con miles de ficheros y miles de líneas de código, con lo que encontrar las líneas a cambiar me costó más de lo que esperaba.

Tras realizar los cambios y ver que todo funcionaba correctamente, mis compañeros me dieron el visto bueno para que subiera los cambios a la rama de *"develop"*, donde todos subíamos las nuevas implementaciones, y que, posteriormente, tenían que pasar por varias ramas más de testeo para finalmente lanzar los cambios a producción.

Por otra parte, una vez terminé la formación en React, no solamente se me asignaban tareas de diseño web, sino que seguía realizando issues sobre las aplicaciones Windows, pero cada una iba suponiendo mayor dificultad y por lo tanto mayor reto que las anteriores.

2.5.3. Podium-Media

Podium-Media marca un antes y un después entre fases durante mi estancia en prácticas porque es la primera gran issue que tuve que realizar en WPF. Este programa, que ya estaba implementado, tenía el simple funcionamiento de mostrar los nombres, y las banderas de los pilotos que habían terminado en pódium en una carrera, donde la bandera representaba las nacionalidades de los tres integrantes del pódium.

La aplicación era bastante simple y mi tarea era muy específica. En primer lugar, tenía que crear una nueva funcionalidad que, dependiendo de la nacionalidad del primer piloto, pudiera sonar el himno de su país al mismo tiempo que se reproducía la animación de las banderas. La funcionalidad era bastante clara y simple, pero tuve que investigar varias cosas a la hora de su implementación.

Esta aplicación, muestra en una gran pantalla de leds, las imágenes producidas que podemos ver en la Figura 9, donde mostramos el nombre de los pilotos, la bandera de su país y la categoría.

La segunda funcionalidad, relacionada directamente con la primera, se trataba de una nueva ventana de ajustes que podemos ver en la Figura 10, la cual permitía al usuario la opción de que el himno sonara automáticamente al iniciar la reproducción o no. Por supuesto si escogía que no, podía igualmente reproducirlo luego a su gusto. Dentro de esta misma ventana, también había una opción de escoger si quería que el himno sonado fuera el del país del piloto en primera posición, o se abría una ventana Windows que le permitía escoger una canción de su propio sistema.

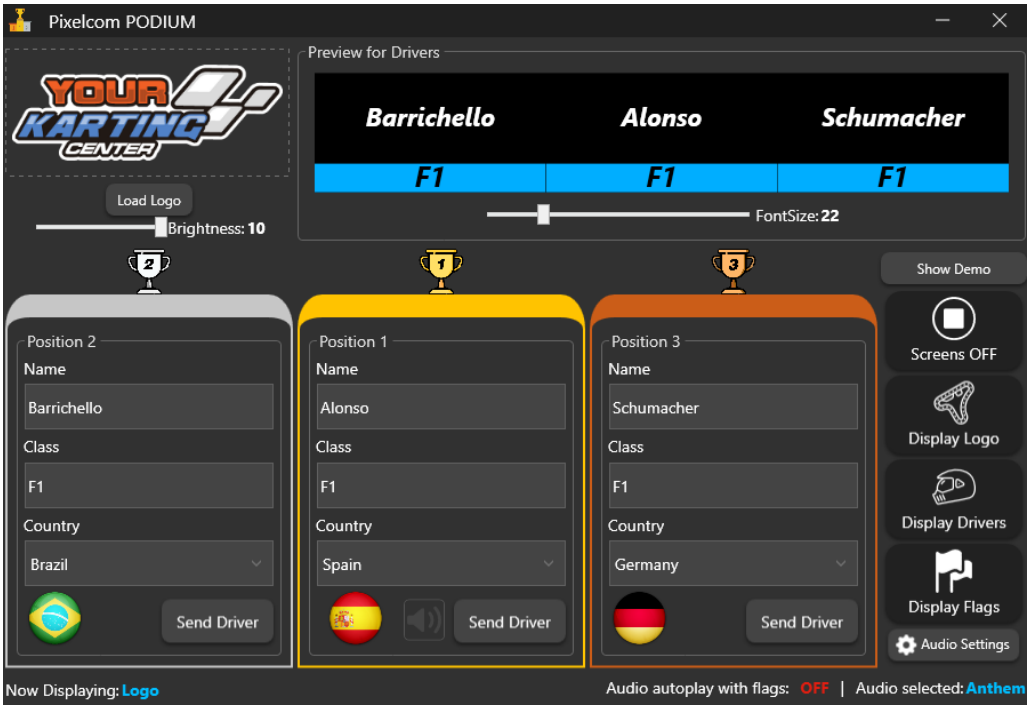


Fig. 9: Ventana principal Podium-Media.

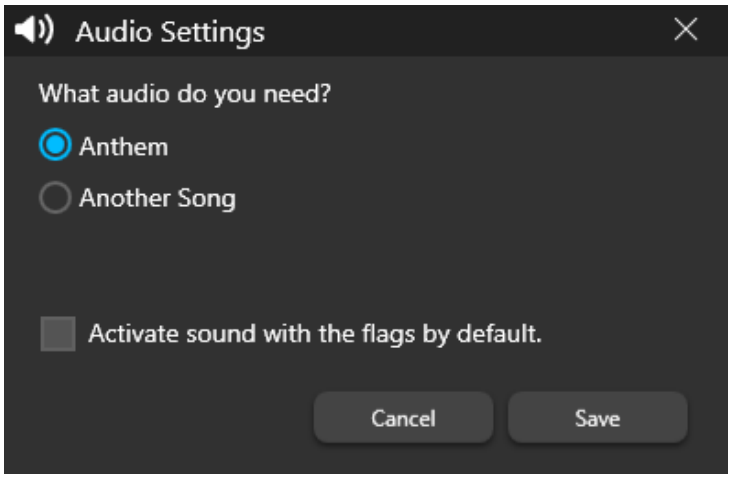


Fig.10 : Ventana de ajustes Podium-Media.

La aplicación como hemos visto era bastante simple, pero pasé de escribir 5 o 6 líneas de código a prácticamente 600 líneas, diferencia que es bastante notable. Además, antes de subir los cambios, un compañero me prohibió subir el código tal y como estaba, porque, básicamente,

reescribía código demasiadas veces y empeoraba tanto la lectura del mismo, como la eficiencia de la aplicación. Claramente tenía razón, porque conseguí reducir aproximadamente 100 líneas de código después de realizar la refactorización del mismo.

Además de esta tarea, realicé algunas más durante este periodo tanto de WPF como de React, pero he decidido expresar esta como ejemplo del cambio de nivel.

2.5.4. React-Native

La parte de React-Native no estaba planificada para el proyecto, pero fue sin duda de las que más me interesaron. El contexto se puede resumir brevemente en que la empresa necesitaba sacar una aplicación a mercado, y debido a los conocimientos que tenía el equipo de desarrollo en lenguaje React, se decidió apostar por una tecnología en auge llamada React-Native, la cual provenía también de Facebook e Instagram. Dicha tecnología se desarrollaba en JavaScript, por lo que no iban a suponer una gran carga para tener que desarrollar la aplicación tanto en Android como en iOS, ya que, esta nueva tecnología permitía escribir aplicaciones nativas para ambos sistemas móviles.

Se organizó un curso intensivo de 3 días, durante los cuales una empresa navarra se dedicaría a enseñarnos todo lo esencial para realizar una aplicación gracias a su experiencia en este sector. Como yo era el más inexperto de todo el equipo de desarrollo, se me permitió una semana antes empezar a realizar tutoriales y a ir experimentando por mi cuenta sobre React-Native para que, llegado el día del curso, pudiera aprovecharlo al máximo.

La verdad es que al haber aprendido previamente React, no suponía el mismo esfuerzo programar en React-Native que si hubiera empezado de nuevo a programar nativo en Android y en iOS. El curso fue bastante bien y, tal y como estaba previsto, aprendimos toda la base de React-Native y todo lo esencial que requería una aplicación, además de cosas que les pedíamos dirigidas específicamente para nuestro proyecto. Luego del curso, empezamos a repartir nuevas tareas entre todo el equipo de desarrollo y estas, iban más enfocadas a la creación de la app.

En la siguiente imagen, podemos ver un ejemplo de aplicación básica. Creamos una función que devuelve una vista ($View$) y dentro tenemos dos componentes básicos, un $Button$ y un $Text$.

Fig. 11: Ejemplo de código en React-Native y su vista final.

2.5.5. Final de la Estancia

Llegados a este punto, llevaría aproximadamente 4-5 meses en la empresa, y para finalizar, empezaría lo que sería una de las grandes tareas a las que me enfrenté durante este proyecto, tarea la cual estuve realizando durante dos semanas.

Al igual que en el apartado del Podium-Media donde hice varias nuevas incorporaciones a la aplicación, se trataba esta vez de lo mismo pero dirigido al diseño web, es decir, para terminar de asentar los conocimientos adquiridos en lenguaje React. Esta vez, la tarea trataba de implementar un nuevo módulo a la aplicación, que debía permitir al usuario visualizar, añadir, editar y borrar una serie de plantillas compuestas por una serie de campos como el nombre, apellidos, DNI, etc. La finalidad de estos campos no se trataba de rellenarlos, sino de activarlos o desactivarlos mediante dos "*checks*", uno que permitía visualizarlo y otro que permitía marcarlo o no como obligatorio a rellenar.

La finalidad de este módulo era permitir a los usuarios (que en este caso eran los gerentes de cualquier Karting), escoger en otra aplicación dirigida de registro de socios, los campos que querían que fueran visibles y de los visibles, los que querían que fueran obligatorios. Cuando vas a un Karting necesitas registrarte y rellenar unos campos para ser socio y poder echar unas carreras, pues eran estos campos los que mi módulo debía de permitir modificar. Además, debía de soportar distintas plantillas de estos campos para cada idioma disponible del karting, con lo que para cada idioma podías tener unos u otros campos marcados como visibles y obligatorios.

Fue la tarea más pesada que tuve que realizar durante toda la estancia en prácticas, ya que, como digo, parece algo simple de primeras, pero el flujo *Redux-Saga* que tuve que entender a la perfección para poder realizar esta tarea fue bastante complejo. El flujo *Redux-Saga*, que solo se usaba en esta aplicación, se trata de un flujo por el que pasa la aplicación cada vez que realiza una acción dentro de ella, y que tiene que pasar por varios ficheros para hacer funcionar dicha acción.

En primer lugar, empezamos por la vista principal, esta es, una vista dentro de la cual insertaremos todos los componentes necesarios como *Buttons, Data-Grid, Toolbar*, etc.

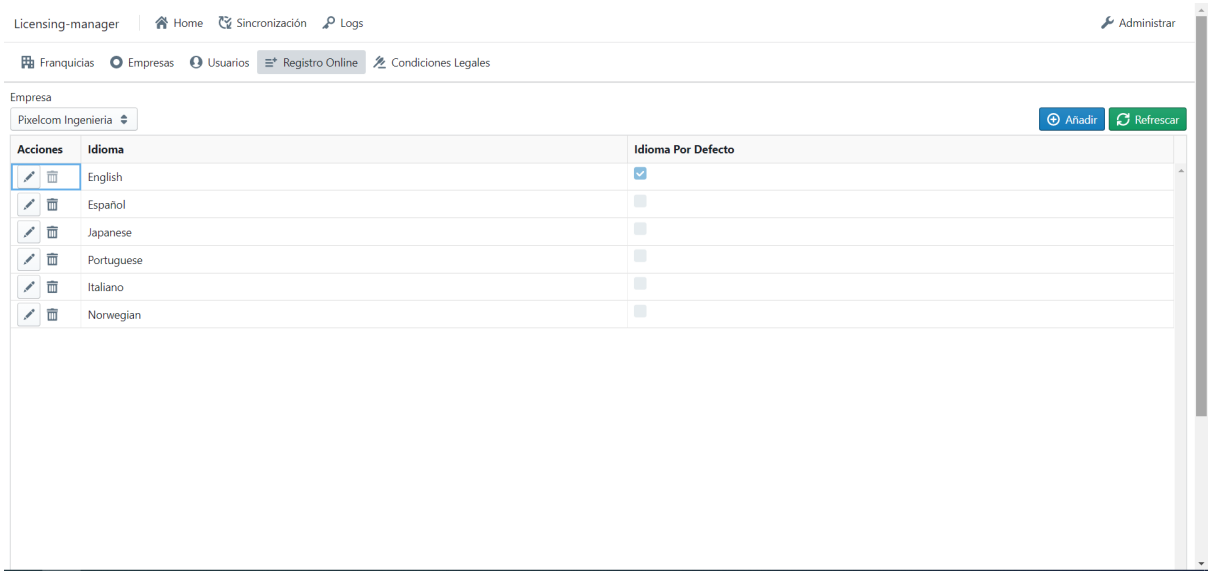


Fig. 12: Vista principal del módulo de Registro Online.

Separamos esta vista en tres componentes principales, el primero es el *Toolbar*, el cual está en cada página de la aplicación por la que viajas. Este te permite seleccionar un filtro por empresa, refrescar la página y añadir nuevas plantillas.

Fig. 13: Toolbar.

En segundo lugar, tenemos la parte inferior, en donde se encuentra la tabla. Esta tabla contiene información sobre los idiomas de los que dispone la empresa y los cuales ya tienen

una plantilla configurada. A la derecha vemos que solo hay un idioma marcado, este es el por defecto de esa empresa. Además, a la parte de la izquierda, podemos apreciar como hay dos componentes *Button*, los cuales permiten tanto editar los campos de la plantilla para ese idioma, como borrarlos.

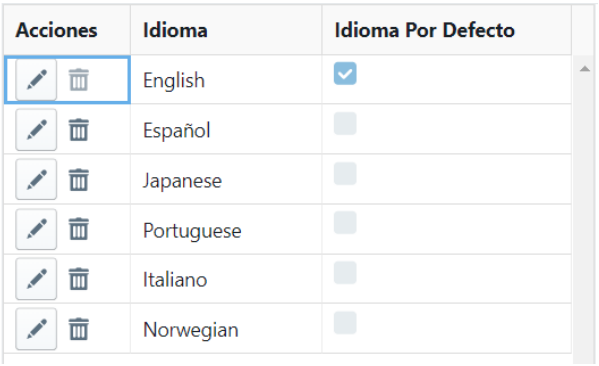


Fig. 14: Tabla de plantillas - *DataGrid*.

Finalmente, la última sub-vista, se trata de la ventana modal que aparece al pulsar tanto en el *Button* de añadir plantilla como de editarla. Dentro de esta vemos todos los campos posibles a editar para un posterior registro.

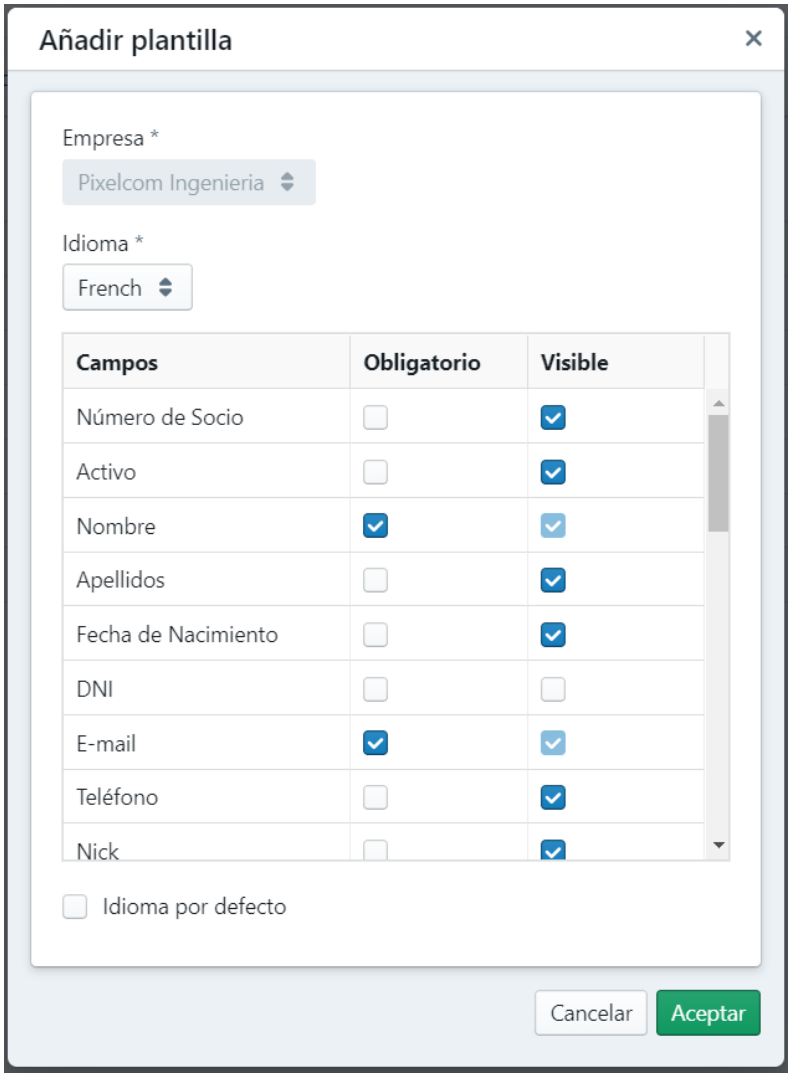


Fig. 15: Ventana modal - Añadir Plantilla.

En cuanto al código, se repite el mismo flujo que tiene que realizar tras la interacción con cualquier componente. Obviamente, cada componente activa unas funciones u otras, pero el flujo de ficheros que tiene que seguir es el mismo, con lo que bastará con un solo ejemplo, que será el de añadir una nueva plantilla.

Primero, tras hacer *Click* sobre el *Button* Añadir, se activa la función *onOpenAddTemplateModal*, la cual está incluida en el *toolbar* del componente *TableTemplates*.

```
107     return (
108       <div>
109         <Navbar location={location} />
110         <div>
111           <NavbarAdmin location={location} />
112         </div>
113
114         <div style={{ margin: '10px' }}>
115           <TableTemplates
116             onEdit={onOpenEditTemplateModal}
117             onDelete={onDeleteTemplate}
118             onRefresh={onRetrieveCurrentTemplates}
119             loading={false}
120             templates={templates}
121             selectedCompany={selectedCompany}
122             toolbar={toolbar(
123               companies,
124               templates,
125               selectedCompany,
126               onFilterTemplatesByCompany,
127               onRetrieveCurrentTemplates,
128               onOpenAddTemplateModal,
129             )}
130           />
131         </div>
132       </div>
133     );
```

Fig. 16: Componente TableTemplates que incluye varias funciones.

Tras reconocer esta función, el flujo se dirige directamente a su acción, pasándole como parámetro el campo *templates*, el cual hemos inyectado previamente con los campos que debe de tener la ventana modal, descritos con anterioridad. Luego, este le redirige hacia el fichero de Acciones, donde como vemos en la Figura 18, se le pasan argumentos en el *return*, entre los cuales vemos un tipo, que será clave.

```
40     onOpenAddTemplateModal: templates => {
41       dispatch(openAddTemplateModalAction(templates));
42     },
```

Fig. 17: Redirección de la función hacia su acción.

```
118  export function openAddTemplateModalAction(objectId) {
119    return {
120      type: OPEN_ADD_TEMPLATE_MODAL,
121      title: 'Añadir plantilla',
122      objectId,
123    };
124  }
```

Fig. 18: Asignación de campos y tipo de acción.

Este, gracias al tipo de acción (*type*), sabe cuál es el componente a devolver, que en nuestro caso es la ventana modal de Añadir plantilla.

```
61      case OPEN_ADD_TEMPLATE_MODAL:
62        return <AddTemplate templates={objectId} />;
```

Fig. 19: Renderización del componente seleccionado.

El estado actual al que ha pasado nuestra vista, se puede apreciar en la Figura 15, vista anteriormente.

Para finalizar el proceso, rellenamos unos cuantos campos o simplemente dejamos los que ya hay por defecto que ya hay, y que previamente habrá configurado el usuario. Nos dirigimos entonces a pulsar Aceptar para que se guarden los cambios. Acto seguido, este será el recorrido de $Redux - Saga$ que realiza la aplicación:

Se activa el campo aceptar tras pulsarlo, el cual como vemos en la siguiente imagen, es del tipo *"submit"*.

```
256                 <BottomDiv>
257                   <Button
258                     intent="none"
259                     style={{ marginRight: '5px' }}
260                     text="Cancelar"
261                     onClick={() => {
262                       onReset();
263                       onCloseModal();
264                     }}
265                   />
266                   <Button
267                     type="submit"
268                     intent="success"
269                     text="Aceptar"
270                     loading={isSubmiting}
271                   />
272                 </BottomDiv>
```

Fig. 20: Componente habitual con el par Aceptar-Cancelar.

Al ser de tipo *"submit"*, se activa la función *onSubmit*, la cual crea un nuevo objeto con los valores actuales que tiene la ventana modal, y acto seguido, la inyecta en otra función *onAddTemplate*.

```
105               onSubmit={(values, { setSubmitting }) => {
106                 setSubmitting(true);
107                 const newTemplate = createTemplate(values);
108                 onAddTemplate(newTemplate);
109               }}
110             >
```

Fig. 21: Función *onSubmit*.

```
32  const createTemplate = values => {
33    template = [];
34    values.currentTemplate.forEach(object => {
35      const data = object.data_member_languages[0];
36      const newObject = {
37        culture: values.selectedLanguage.culture,
38        idCompany: data.idCompany,
39        idDataMember: data.idDataMember,
40        idKioskPage: data.idKioskPage,
41        idLocal: data.idLocal,
42        required: data.required,
43        visible: data.visible,
44      };
45      template.push(newObject);
46    });
47
48    const newTemplate = {
49      template,
50      isDefault: values.defaultTemplateCheck,
51      idCompany: values.company.id,
52    };
53    return newTemplate;
54  };
```

Fig. 22: Función *createTemplate*.

Acto seguido, sigue el flujo por una función que al igual que la de abrir modal, se redirige hacia su correspondiente acción, que en este caso es el de crear la plantilla.

```
35    onAddTemplate: template => {
36      dispatch(createTemplateAction(template));
37      dispatch(retrieveCurrentTemplatesAction(template.idCompany));
38    },
```

Fig. 23: Función que redirige a la acción de crear plantilla.

```
17  export function createTemplateAction(template) {
18    return {
19      type: CREATE_TEMPLATE,
20      template,
21    };
22  }
```

Fig. 24: Acción de crear plantilla.

En este punto, a diferencia de devolver un componente, nuestra función tiene que crear la plantilla, por lo tanto, redirige la acción al archivo de saga. Aquí, dependiendo del tipo que le hayamos dado, ejecutará una u otra función, que, en nuestro caso, será *createTemplateSaga*.

```
36  export default function* addTemplateSaga() {
37    yield takeLatest(CREATE_TEMPLATE, createTemplateSaga);
38    yield takeLatest(RETRIEVE_PENDING_LANGUAGES, retrievePendingLanguagesSaga);
39  }
```

Fig. 25: Redirección del flujo dependiendo del campo *type*.

```
16  export function* createTemplateSaga({ template }) {
17    try {
18      const response = yield call(createTemplate, template);
19      yield put(createTemplateSuccessAction(response));
20      yield put(closeModalAction());
21      yield put(retrieveCurrentTemplatesAction(template.idCompany));
22    } catch (err) {
23      yield put(createTemplateErrorAction(err));
24    }
25  }
```

Fig. 26: Función Saga de crear plantilla.

Una vez aquí, lo primero es esperar a que se complete la variable *response*, la cual lanza otra función, que conecta nuestra aplicación con otro servicio destinado íntegramente a la gestión directa con la base de datos. Al llamar a este servicio, lo que hacemos es enviarle el objeto que previamente hemos construido, para así guardarse esta nueva información.

```
9   export const createTemplate = async template => {
10    try {
11      const response = await axios.post(
12        `${configJs.baseUrl}/data-member-language/`,
13        template,
14        {
15          headers: httpReqHeaders,
16        },
17      );
18      return response.data;
19    } catch (error) {
20      throw error.response.status;
21    }
22  };
```

Fig. 27: Llamada a otro servicio.

El servicio nos devuelve una respuesta, que en nuestro caso está vacía (si en vez de crear algo, fuera pedir algo, nos devolvería información). Al recibir la respuesta, el flujo vuelve a la función saga (Fig. 26), donde vemos que le siguen 3 funciones. La primera, *createTemplateSuccessAction* devuelve la respuesta a modo de estado, el cual actualiza las variables de la siguiente Figura 29. La segunda, *closeModalAction*, tal y como indica, su función es cerrar la ventana modal, puesto que ya hemos creado la plantilla; y la tercera, llamada *retrieveCurrentTemplatesAction*, recurre a la acción de dicha función, cuya finalidad no es otra que la de refrescar la página automáticamente tras crear la plantilla, para así visualizar los cambios en nuestra tabla.

```
36      case RETRIEVE_CURRENT_TEMPLATES_SUCCESS:
37        return state
38          .set('loading', false)
39          .set('error', false)
40          .set('templates', action.templates);
```

Fig. 28: Actualización del estado.

En la Figura 28, vemos que se actualiza el estado de la aplicación, quitando el estado de carga (*loading*), el de error, y el campo *templates*, se actualiza con el nuevo *action.templates*, el cual incluye la plantilla que acabamos de añadir.

2.6. Conclusiones

La verdad es que mi estancia en prácticas ha sido más satisfactoria de lo que pensaba. He tenido durante toda la estancia una relación muy cercana con cada uno de los compañeros de la empresa, tanto del departamento de desarrollo, como el de soporte informático, comerciales, etc.

He aprendido durante la estancia muchas cosas que no se reducen solo a mejorías de programación y de lenguajes, sino que he desarrollado una mayor capacidad a la hora de realizar trabajo en equipo, ya que las tareas asignadas lo requerían. Además, creo que he obtenido una muy buena formación, sobre todo porque mis compañeros del departamento de desarrollo pasaron por mi misma situación en su momento y he recibido mucha ayuda por su parte.

Pienso que mi desarrollo en la programación solo acaba de empezar. Además, al terminar mi estancia en prácticas, se me ofreció un contrato laboral en Pixelcom y sigo trabajando aquí a día de hoy, donde he aprendido y sigo aprendiendo muchos más conocimientos de los que aprendí durante la estancia, pero no tienen cabida en esta memoria.

Capítulo 3

Memoria TFG

3.1. Motivación y Objetivos

El estudio realizado sobre los modelos de aguas someras (shallow water en la literatura anglosajona), tiene como objetivo desarrollar y comprender los fenómenos físicos que tienen lugar en aguas poco profundas, mediante un conjunto de ecuaciones diferenciales en derivadas parciales de carácter hiperbólico. Dicho fenómeno físico tiene lugar en los movimientos de flujos de agua como pueden ocurrir en las mareas, en las corrientes fluviales en ríos, lagos, en movimientos más bruscos como cuando rompen las olas, el movimiento ondulante de las olas en canales abiertos, roturas de presas e incluso tsunamis.

3.2. Ecuaciones que gobiernan el flujo de agua

Las ecuaciones de shallow water que vamos a considerar en esta sección, van a depender de dos dimensiones espaciales y de la variable tiempo. El objetivo es describir la evolución del fenómeno físico que se produce en un dominio $(x, y) \in \mathbf{R}^2$ cuando evoluciona $t \geq 0$. Mostraremos que las condiciones del fenómeno físico nos conducen a un sistema de tres ecuaciones diferenciales en derivadas parciales no lineales, donde una de las incógnitas es la profundidad del agua. Esto nos va a permitir visualizar el fenómeno en tres dimensiones.

En primer lugar, precisamos introducir dos principios de conservación, esenciales en este fenómeno físico, la ley de conservación de la masa y la ley de conservación del momento, lo cual vamos a detallar en la siguiente sección.

3.2.1. Leyes de la conservación

Las ecuaciones de aguas someras derivan de las ecuaciones de los dos principios mencionados anteriormente, es decir, de las ecuaciones de conservación de la masa y de las ecuaciones de conservación del momento. Podemos elegir entre dos modelos para justificar nuestro estudio: el modelo de Euler (fluido no viscoso) y el modelo de Navier-Stokes (fluido con viscosidad). En este trabajo nos vamos a decantar por el primero de ellos.

Para un estudio más general, consideraremos en un principio tres dimensiones espaciales y las ecuaciones de conservación de la masa y del momento siguientes:

$$\rho_t + \nabla \cdot (\rho V) = 0, \tag{3.1}$$

y

$$\frac{\partial}{\partial t}(\rho V) + \nabla \cdot [\rho V \otimes V + pI - \mathfrak{T}] = \rho g \tag{3.2}$$

Donde aparecen las variables independientes t para el tiempo y (x, y, z) para el espacio. El resto de variables (y constantes en algunos casos) serán V = (u, v, w) la velocidad, ρ la densidad, p la presión y g el vector gravedad $g = (g_1, g_2, g_3)$. Además, intervienen, el vector gradiente ∇ a V, el tensor V $\otimes$ V producto tensorial del vector velocidad, la matriz identidad I, y el tensor de tensión viscosa $\mathfrak{T}$.

Veámoslo de una manera más detallada:

$$V \otimes V = \begin{pmatrix} u^2 & uv & uw \\ vu & v^2 & vw \\ wu & wv & w^2 \end{pmatrix}, \; I = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \; \mathfrak{T} = \begin{bmatrix} \tau^{xx} & \tau^{xy} & \tau^{xz} \\ \tau^{yx} & \tau^{yy} & \tau^{xz} \\ \tau^{zx} & \tau^{zy} & \tau^{zz} \end{bmatrix}.$$

A partir de las ecuaciones (3.1) y (3.2), consideraremos el modelo Euler, donde la masa se conserva y los efectos de la viscosidad del fluido son nulos.

Utilizando la ecuación de conservación de la masa (3.1): $\rho_t + \nabla \cdot (\rho V) = 0$, obtenemos

$$\rho_t + (u\frac{\partial \rho}{\partial x} + v\frac{\partial \rho}{\partial y} + w\frac{\partial \rho}{\partial z}) + (\rho\frac{\partial u}{\partial x} + \rho\frac{\partial v}{\partial y} + \rho\frac{\partial w}{\partial z}) = 0$$

$$\rho_t + u\rho_x + v\rho_y + w\rho_z + \rho(u_x + v_y + w_z) = 0$$

Con objeto de simplificar las ecuaciones, vamos a suponer que la densidad del líquido es siempre constante al variar el tiempo. Esto, aplicado a la conservación de la masa, significa que la divergencia de la velocidad de nuestro fluido es cero, considerando como definición de divergencia, el campo vectorial que mide la diferencia entre el flujo saliente y el entrante, en un campo vectorial de una superficie. Dicho esto, el desarrollo de la ecuación de la conservación de la masa queda de la siguiente forma:

$$\rho_t + u\rho_x + v\rho_y + w\rho_z + \rho(u_x + v_y + w_z) = \rho(u_x + v_y + w_z) = 0$$

Finalmente:

$$u_x + v_y + w_z = 0 \tag{3.3}$$

Ahora desarrollaremos de la misma manera las ecuaciones de conservación del momento. Partiendo de la ecuación (3.2): $\frac{\partial}{\partial t}(\rho V) + \nabla\cdot[\rho V \otimes V + pI - \mathfrak{T}] = \rho g$, donde, considerando a un fluido no viscoso ($\mathfrak{T} = 0$), obtenemos:

$$\frac{\partial}{\partial t}(\rho V) + \nabla\cdot[\rho V \otimes V + pI] = \rho g$$

Necesitaremos dos propiedades antes de seguir, la primera de ellas es una propiedad de los productos tensoriales de dos vectores, la segunda, como hemos dicho previamente, la divergencia de la velocidad de nuestro fluido es cero:

$$ca \otimes b = a \otimes cb = c(a \otimes b) \tag{3.4}$$

$$div(V) = \nabla \cdot (V) = u_x + v_y + w_z = 0 \tag{3.5}$$

Siguiendo con el planteamiento, y con el tensor de viscosidad nulo:

$$\rho\frac{\partial}{\partial t}(V) + V\frac{\partial}{\partial t}(\rho) + \nabla\cdot[\rho V \otimes V + pI] = \rho g$$

$$\rho\frac{\partial}{\partial t}(V) + V\frac{\partial}{\partial t}(\rho) + \nabla\cdot(\rho V \otimes V) + \nabla\cdot(pI) = \rho g$$

Aplicando (3.4):

$$\rho\tfrac{\partial}{\partial t}(V) + V\tfrac{\partial}{\partial t}(\rho) + \nabla\cdot(\rho(V\otimes V)) + \nabla\cdot(pI) = \rho g$$

$$\rho\tfrac{\partial}{\partial t}(V) + V\tfrac{\partial}{\partial t}(\rho) + \rho\nabla\cdot(V\otimes V) + \nabla\cdot(pI) = \rho g$$

Ahora, hay que tener en cuenta que la densidad ρ es constante, $\dfrac{\partial\rho}{\partial t} = 0$, así

$$\nabla\cdot(\rho) = (\frac{\partial\rho}{\partial x} + \frac{\partial\rho}{\partial y} + \frac{\partial\rho}{\partial z}) = (0+0+0) = 0.$$

Por tanto

$$\rho\frac{\partial}{\partial t}(V) + \rho\nabla\cdot(V\otimes V) + \nabla\cdot(pI) = \rho g,$$

De la ecuación (3.4) se deduce en [1] lo siguiente:

$$\nabla\cdot(V\otimes V) = V\cdot\nabla(V)$$

Pero, a continuación, vamos a comprobar esta igualdad. Tenemos en (3.5) que $\nabla\cdot V = u_x + v_y + w_z = 0$, así:

$$V\cdot\nabla(V) = \begin{pmatrix} u & v & w \end{pmatrix}\cdot\begin{pmatrix} u_x & v_x & w_x \\ u_y & v_y & w_y \\ u_z & v_z & w_z \end{pmatrix} = \begin{pmatrix} uu_x + vu_y + wu_z \\ uv_x + vv_y + wv_z \\ uw_x + vw_y + ww_z \end{pmatrix}$$

Vamos a ver si coincide con $\nabla\cdot(V\otimes V)$:

$$\nabla\cdot(V\otimes V) = (\tfrac{\partial}{\partial x},\tfrac{\partial}{\partial y},\tfrac{\partial}{\partial z})\cdot\begin{pmatrix} u^2 & uv & uw \\ vu & v^2 & vw \\ wu & wv & w^2 \end{pmatrix} = \begin{pmatrix} 2uu_x + v_yu + vu_y + w_zu + wu_z \\ u_xv + uv_x + 2vv_y + v_zw + vw_z \\ u_xw + uw_x + v_yw + vw_y + 2ww_z \end{pmatrix} =$$

$$\begin{pmatrix} uu_x + vu_y + wu_z + [u(u_x + v_y + w_z)] \\ uv_x + vv_y + vw_z + [v(u_x + v_y + w_z)] \\ uw_x + vw_y + ww_z + [w(u_x + v_y + w_z)] \end{pmatrix} =$$

Aplicamos (3.5):

$$\begin{pmatrix} uu_x + vu_y + wu_z \\ uv_x + vv_y + vw_z \\ uw_x + vw_y + ww_z \end{pmatrix}$$

Con lo que se cumple que $\nabla \cdot (V \otimes V) = V \cdot \nabla(V)$

Por último, nos queda el término $\nabla \cdot (pI)$:

$$\nabla \cdot (pI) = \nabla \cdot p \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \nabla \cdot \begin{pmatrix} p & 0 & 0 \\ 0 & p & 0 \\ 0 & 0 & p \end{pmatrix} = (\tfrac{\partial}{\partial x}, \tfrac{\partial}{\partial y}, \tfrac{\partial}{\partial z}) \cdot \begin{pmatrix} p & 0 & 0 \\ 0 & p & 0 \\ 0 & 0 & p \end{pmatrix} = (p_x, p_y, p_z)$$

Finalmente, nuestra ecuación quedará de la siguiente forma:

$$\rho \frac{\partial}{\partial t}(V) + \rho V \cdot \nabla(V) + \nabla(pI) = \rho g,$$

$$\rho \frac{\partial}{\partial t}((u,v,w)) + \rho(u,v,w) \cdot \nabla((u,v,w)) + \nabla(pI) = \rho g,$$

$$\rho(\frac{\partial u}{\partial t}, \frac{\partial v}{\partial t}, \frac{\partial w}{\partial t}) + \rho(u,v,w)((\frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}, \frac{\partial u}{\partial z}), (\frac{\partial v}{\partial x}, \frac{\partial v}{\partial y}, \frac{\partial v}{\partial z}), (\frac{\partial w}{\partial x}, \frac{\partial w}{\partial y}, \frac{\partial w}{\partial z})) + (\frac{\partial p}{\partial x}, \frac{\partial p}{\partial y}, \frac{\partial p}{\partial z}) = \rho g,$$

$$(\frac{\partial u}{\partial t}, \frac{\partial v}{\partial t}, \frac{\partial w}{\partial t}) + (u,v,w)((\frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}, \frac{\partial u}{\partial z}), (\frac{\partial v}{\partial x}, \frac{\partial v}{\partial y}, \frac{\partial v}{\partial z}), (\frac{\partial w}{\partial x}, \frac{\partial w}{\partial y}, \frac{\partial w}{\partial z})) + \frac{(\frac{\partial p}{\partial x}, \frac{\partial p}{\partial y}, \frac{\partial p}{\partial z})}{\rho} = (g_1, g_2, g_3).$$

Por comodidad, cambiamos la notación, así

$$(u_t, v_t, w_t) + (u,v,w)((u_x, u_y, u_z), (v_x, v_y, v_z), (w_x, w_y, w_z)) + \tfrac{(p_x, p_y, p_z)}{\rho} = (g_1, g_2, g_3)$$

$$(u_t, v_t, w_t) + (u,v,w)(u_x, u_y, u_z) + (u,v,w)(v_x, v_y, v_z) + (u,v,w)(w_x, w_y, w_z) + \tfrac{(p_x, p_y, p_z)}{\rho} = (g_1, g_2, g_3)$$

De esta manera, nos quedan tres ecuaciones para las tres componentes espaciales (x, y, z) :

$$\left.\begin{array}{c} u_t + uu_x + vu_y + wu_z + \frac{1}{\rho}p_x = g_1 \\ v_t + uv_x + vv_y + wv_z + \frac{1}{\rho}p_y = g_2 \\ w_t + uw_x + vw_y + ww_z + \frac{1}{\rho}p_z = g_3 \end{array}\right\}$$

Estas ecuaciones, junto con la ecuación (3.3), forman las ecuaciones necesarias con las que posteriormente, definiremos el problema de superficie libre. En resumen:

$$\left.\begin{array}{c} u_x + v_y + w_z = 0 \\ u_t + uu_x + vu_y + wu_z + \frac{1}{\rho}p_x = g_1 \\ v_t + uv_x + vv_y + wv_z + \frac{1}{\rho}p_y = g_2 \\ w_t + uw_x + vw_y + ww_z + \frac{1}{\rho}p_z = g_3 \end{array}\right\}$$

3.2.2. Flujo de agua en superficie libre

El flujo de agua en una superficie libre esta gobernado por un sistema hiperbólico de leyes de conservación. Las magnitudes conservadas son, en nuestro caso, la masa y el momento. Dicho sistema hiperbólico, recibe el nombre ecuaciones de aguas someras (ecuaciones shallow water en la literatura anglosajona y ecuaciones de Saint-Venant en la literatura francófona).

Para representar una superficie libre, hemos de definir varias funciones. En primer lugar, el fondo de la superficie al que denotaremos por z, será representado por una función de dos variables $z = b(x, y)$. La superficie libre, se define como: $z = s(x, y, t) \equiv b(x, y)+h(x, y, t)$. Donde $h(x, y, t)$ es la distancia entre el fondo y la superficie, o dicho de otra forma, la profundidad del agua. En la siguiente imagen, apreciamos estas definiciones de forma más clara.

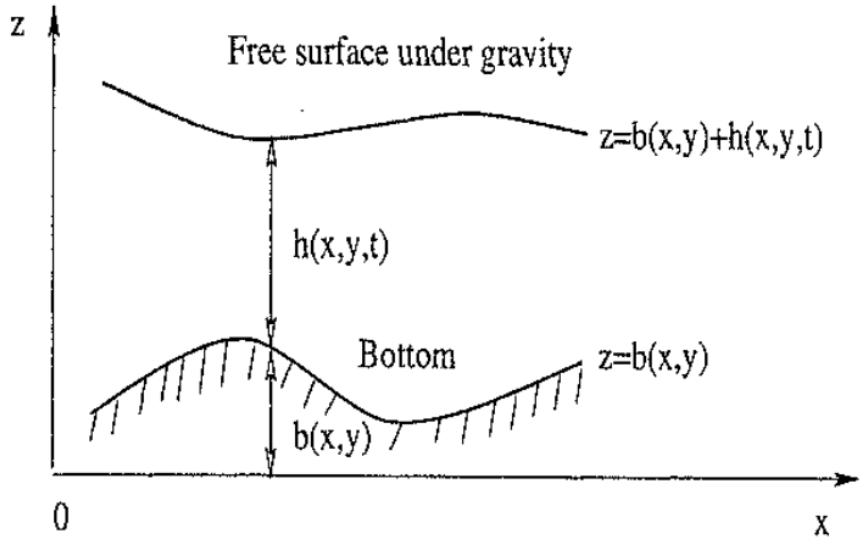


Fig. 29: Flujo en una superficie libre bajo gravedad, para plano y fijo.
Shock-Capturing Methods for Free-Surface Shallow Flows, Eleuterio F. Toro

Ahora vamos a suponer que las fuerzas que actúan sobre el fluido únicamente dependen de la gravedad, con lo cual, la gravedad afecta verticalmente al eje z de forma negativa, así, $g = (g_1, g_2, g_3) = (0, 0, -g)$. Consideraremos que la aceleración debida a la gravedad, viene dada por el valor de $g = 9{,}8\, m/s^2$.

Por lo tanto, las ecuaciones obtenidas previamente, quedan de la siguiente forma:

$$u_x + v_y + w_z = 0 \tag{3.6}$$

$$u_t + uu_x + vu_y + wu_z = -\frac{1}{\rho}p_x \tag{3.7}$$

$$v_t + uv_x + vv_y + wv_z = -\frac{1}{\rho}p_y \tag{3.8}$$

$$w_t + uw_x + vw_y + ww_z = -\frac{1}{\rho}p_z - g \tag{3.9}$$

3.2.3. Condiciones de frontera previas a las ecuaciones

Previamente a la obtención las ecuaciones shallow water, se deben obtener las ecuaciones de superficie libre. Es decir, las condiciones de frontera para las ecuaciones (3.4)-(3.7).

Supongamos que la frontera viene dada por la superficie libre: $\psi(x, y, z, t) = 0$. Así,

$$\psi(x, y, z, t) \equiv z - s(x, y, t) = 0. \tag{3.10}$$

Las condiciones para el fondo serán:

$$\psi(x, y, z, t) \equiv z - b(x, y) = 0. \tag{3.11}$$

Ahora, en la superficie libre $s(x, y, t)$, consideramos las siguientes dos condiciones de frontera, llamadas condición cinemática y condición dinámica. La condición cinemática se define como:

$$\frac{\partial}{\partial t}\psi(x, y, z, t) \equiv \psi_t + u\psi_x + v\psi_y + w\psi_z = 0. \tag{3.12}$$

Análogamente, la condición dinámica:

$$p(x, y, z, t)|_{z=s(x,y)} = p_{atm} = 0. \tag{3.13}$$

Es decir, la presión cuando se encuentra a la altura de la superficie libre, se trata de la presión atmosférica, que, por simplicidad, la tomamos como nula.

3.2.4. Las ecuaciones de Shallow Water

En primer lugar, vamos a suponer que la aceleración en dirección vertical es despreciable, es decir, $\dfrac{\partial w}{\partial t} = w_t + uw_x + vw_y + ww_z$. Así, $\frac{\partial w}{\partial t} = 0$. Entonces, introduciendo esta condición en la ecuación (3.9), obtenemos: $0 = -\frac{1}{\rho}p_z - g$, $\frac{1}{\rho}p_z = -g, p_z = -\rho g$.

Utilizando la condición (3.13), sabiendo que la presión es nula en la superficie, con $z = s(x, y)$ y $p = 0$ e integrando, obtenemos la igualdad:

$$\int_z^s -p_z d_z = \int_z^s \rho g d_z.$$

Calcularemos primero la integral de la izquierda. Para $z = s(x, y)$, siendo s la superficie del agua en el punto más alto, y z cualquier otro punto arbitrario dentro del agua. Entonces

calculando la primera integral, obtenemos:

$$\int_z^s -p_z d_z = [-p]_z^s = -p.$$

Aplicando la ecuación (3.13), nos queda $-p = 0 = p_{atm} = p$.

Ahora, sustituyendo el resultado en la igualdad principal:

$$p = \int_z^s \rho g d_z = 0, \tag{3.14}$$

resolviendo la integral:

$$\int_z^s \rho g d_z = \rho g \int_z^s d_z = \rho g [z]_z^s = \rho g (s - z),$$

aplicando (3.12): $\rho g(s - z) = 0$. Por lo tanto, tras integrar, nos queda la siguiente igualdad:

$$p = \rho g (s - z). \tag{3.15}$$

Derivando (3.13) en x y en y, obtenemos:

$$p_x = \rho g s_x, \;\; p_y = \rho g s_y. \tag{3.16}$$

Las ecuaciones que acabamos de obtener nos muestran que tanto p_x como p_y son ambas independientes de z, y, por lo tanto, las componentes de la aceleración del agua $\frac{\partial u}{\partial t}$ y $\frac{\partial v}{\partial t}$, son también independientes de z. Entonces, los componentes velocidad u y v son independientes de z, es decir, $u_z = v_z = 0$. Por lo tanto, si observamos la ecuación (3.7): $u_t + uu_x + vu_y + wu_z = -\frac{1}{\rho} p_x$, y como sabemos que $u_z = 0$, tenemos $u_t + uu_x + vu_y = -\frac{1}{\rho} p_x$ y sustituyendo la igualdad (3.16) podemos obtener $u_t + uu_x + vu_y = -\frac{1}{\rho} \rho g s_x$ y $v_t + uv_y + vv_y = -g s_y$.

Por lo que hemos obtenido las siguientes ecuaciones:

$$u_t + uu_x + vu_x = -g s_x \tag{3.17}$$

$$v_t + uu_y + vu_y = -g s_y \tag{3.18}$$

El problema principal de la superficie libre se puede resolver mediante el cálculo del área comprendida entre la superficie del agua $z = s(x, y, t)$ y el fondo $z = b(x, y)$, mediante la ecuación de conservación de la masa (3.4), previamente obtenida en los anteriores apartados.

$$\int_b^s (u_x + v_y + w_z)\, d_z = 0 \tag{3.19}$$

$$\int_b^s u_x \, d_z + \int_b^s v_y \, d_z + \int_b^s w_z \, d_z = 0$$

$$\int_b^s u_x \, d_z + \int_b^s v_y \, d_z + w|_{z=s} - w|_{z=b} = 0 \tag{3.20}$$

Aplicando ahora la condición cinemática (3.12), sobre la superficie libre (3.10) tenemos que:

$$\psi(x, y, z, t) \equiv z - s(x, y, t) = 0$$

$$\frac{\partial}{\partial t}\psi(x, y, z, t) = (w\frac{\partial z}{\partial z} - (\frac{\partial s}{\partial t} + u\frac{\partial s}{\partial x} + v\frac{\partial s}{\partial y}))|_{z=s} = 0$$

$$\frac{\partial}{\partial t}\psi(x, y, z, t) = (w(1) - (s_t + us_x + vs_y))|_{z=s} = 0$$

$$w|_{z=s} - (s_t + us_x + vs_y)|_{z=s} = 0$$

Entonces, hemos obtenido la siguiente igualdad:

$$w|_{z=s} = (s_t + us_x + vs_y)|_{z=s} \tag{3.21}$$

Realizamos ahora el mismo procedimiento, aplicando la condición cinemática, esta vez a la ecuación que representa el fondo (3.11).

$$\psi(x, y, z, t) \equiv z - b(x, y) = 0$$

De nuevo, volvemos a aplicar la condición cinemática (3.12):

$$\frac{\partial}{\partial t}\psi(x, y, z, t) = (w\frac{\partial z}{\partial z} - (u\frac{\partial b}{\partial x} + v\frac{\partial b}{\partial y}))|_{z=b} = 0$$

$$\frac{\partial}{\partial t}\psi(x, y, z, t) = (w(1) - (ub_x + vb_y))|_{z=b} = 0$$

$$(w - ub_x - vb_y)|_{z=b} = 0$$

Y tenemos la siguiente igualdad:

$$w|_{z=b} = (ub_x + vb_y)|_{z=b} \tag{3.22}$$

Sustituyendo en la ecuación en la que nos hemos quedado integrando (3.20), nos queda:

$$\int_b^s u_x \, d_z + \int_b^s v_y \, d_z + (s_t + us_x + vs_y)|_{z=s} - (ub_x + vb_y)|_{z=b} = 0 \tag{3.23}$$

A partir de aquí necesitaremos enunciar la fórmula de Leibniz para poder seguir.

$$\frac{d}{dx}\int_{t_1(x)}^{t_2(x)} f(t,x)dt = \int_{t_1(x)}^{t_2(x)} \frac{\partial f}{\partial x}dt + f(t_2,x)\cdot\frac{d}{dx}t_2 - f(t_1,x)\cdot\frac{d}{dx}t_1 \tag{3.24}$$

Aplicamos ahora esta fórmula (3.24) a nuestra ecuación (3.23) por integrales:

$$\int_b^s u_x \, d_z = \frac{\partial}{\partial x}\int_b^s u \, d_z - u|_{z=s}\cdot s_x + u|_{z=b}\cdot b_x \tag{3.25}$$

Igual para la integral restante:

$$\int_b^s v_y \, d_z = \frac{\partial}{\partial y}\int_b^s v \, d_z - v|_{z=s}\cdot s_y + v|_{z=b}\cdot b_y \tag{3.26}$$

Ya tenemos todas las igualdades necesarias, ahora sustituimos en la ecuación (3.23):

$$\int_b^s u_x \, d_z + \int_b^s v_y \, d_z + (s_t + us_x + vs_y)|_{z=s} - (ub_x + vb_y)|_{z=b} =$$

$$\frac{\partial}{\partial x}\int_b^s u \, d_z - u|_{z=s}\cdot s_x + u|_{z=b}\cdot b_x + \frac{\partial}{\partial y}\int_b^s v \, d_z - v|_{z=s}\cdot s_y + v|_{z=b}\cdot b_y +$$

$$(s_t + us_x + vs_y)|_{z=s} - (ub_x + vb_y)|_{z=b} = 0$$

$$\frac{\partial}{\partial x}\int_b^s u \, d_z + \frac{\partial}{\partial y}\int_b^s v \, d_z - u|_{z=s}\cdot s_x + us_x|_{z=s} + u|_{z=b}\cdot b_x - ub_x|_{z=b}$$

$$-v|_{z=s}\cdot s_y + vs_y|_{z=s} + v|_{z=b}\cdot b_y - vb_y|_{z=b} + s_t = 0$$

Simplificando términos, nos queda la siguiente ecuación:

$$s_t + \frac{\partial}{\partial x}\int_b^s u \, d_z + \frac{\partial}{\partial y}\int_b^s v \, d_z = 0 \tag{3.27}$$

Finalmente, como u y v son independientes de z tal como hemos dicho anteriormente (debajo de la ecuación 2.16), como la superficie es equivalente al fondo sumado a la altura ($s = b + h$) simplificaremos la ecuación de la siguiente forma:

$$s_t + \frac{\partial}{\partial x}([uz]_b^s) + \frac{\partial}{\partial y}([vz]_b^s) = 0$$

$$s_t + \frac{\partial}{\partial x}[u(s-b)] + \frac{\partial}{\partial y}[v(s-b)] = 0$$

Sustituimos ($s = b + h$):

$$(b+h)_t + \frac{\partial}{\partial x}[u(b+h-b)] + \frac{\partial}{\partial y}[v(b+h-b)] = 0$$

$$b_t + h_t + \frac{\partial}{\partial x}(hu) + \frac{\partial}{\partial y}(hv) = 0$$

$$b_t + h_t + (hu)_x + (hv)_y = 0$$

Donde $b_t = 0$, por lo tanto, finalmente tenemos la ecuación de forma simplificada:

$$h_t + (hu)_x + (hv)_y = 0 \tag{3.28}$$

Esta, es la ecuación de la ley de la conservación de la masa (3.6) escrita en forma diferencial. Para obtener también de esta forma diferencial las ecuaciones (3.7) y (3.8), necesitaremos multiplicar por u la ecuación (3.28) que acabamos de obtener y por h a las ecuaciones (3.17) y (3.18), que han sido obtenidas a partir del razonamiento anterior de u y v independientes de z. Además, tenemos que hacer uso de la relación que asume la diferenciabilidad de la profundidad del agua h como vemos en [1]:

$$h\frac{\partial h}{\partial x} = \frac{\partial}{\partial x}\left(\frac{1}{2}h^2\right) \tag{3.29}$$

Haremos el razonamiento para x. En primer lugar, tras multiplicar por u y por h las ecuaciones (3.28) y (3.17) respectivamente, obtenemos las siguientes ecuaciones:

$$hu_t + huu_x + hvu_y = -gs_x h \tag{3.30}$$

$$[h_t + (hu)_x + (hv)_y]u = 0 \tag{3.31}$$

Sabiendo que $(hu)_t = h_t u + hu_t$, y despejando de esta ecuación hu_t, obtenemos la siguiente igualdad:

$$hu_t = (hu)_t - h_t u \tag{3.32}$$

Lo haremos también para la ecuación $(hu^2)_x = (hu)_x u + huu_x$, donde en este caso, despejamos huu_x:

$$huu_x = (hu^2)_x - (hu)_x u \tag{3.33}$$

Ahora, tal y como hemos mencionado previamente (antes de la ecuación 3.28), la superficie es equivalente al fondo junto con la altura, $s = b + h$, por lo tanto, vamos con la parte derecha de nuestra ecuación (3.30):

$$-gs_x h = -g(b+h)_x h = -g(b_x + h_x)h = -gb_x h - ghh_x = -gb_x h - \left(g\frac{1}{2}h^2\right)_x \tag{3.34}$$

Por último, al igual que los anteriores pasos, despejaremos hvu_y, de la ecuación $(hvu)_y = (hv)_y u + huv_y$, y queda:

$$hvu_y = (hvu)_y - (hv)_y u \tag{3.35}$$

Sustituyendo las ecuaciones (3.32) - (3.35) en la ecuación (3.30):

$$(hu)_t - h_t u + (hu^2)_x - (hu)_x u + (hvu)_y - (hv)_y u = -gb_x h - (\tfrac{1}{2}gh^2)_x$$

$$(hu)_t + (hu^2)_x + (\tfrac{1}{2}gh^2)_x + (hvu)_y = h_t u + (hu)_x u + (hv)_y u - gb_x h$$

$$(hu)_t + (hu^2)_x + (\tfrac{1}{2}gh^2)_x + (hvu)_y = [h_t + (hu)_x + (hv)_y]u - gb_x h$$

$$(hu)_t + (hu^2 + \tfrac{1}{2}gh^2)_x + (hvu)_y = [h_t + (hu)_x + (hv)_y]u - gb_x h$$

Finalmente, aplicando la ecuación (3.31):

$$(hu)_t + (hu^2 + \frac{1}{2}gh^2)_x + (hvu)_y = -ghb_x \tag{3.36}$$

De la misma forma, obtenemos para el momento y:

$$(hv)_t + (huv)_x + (hv^2 + \frac{1}{2}gh^2)_y + = -ghb_y \tag{3.37}$$

Estas tres ecuaciones obtenidas, (3.28), (3.36) y (3.37) las podemos escribir en su forma diferencial de ley de conservación como una única ecuación vectorial.

En primer lugar, tenemos el vector de variables de la conservación.

$$U = \begin{bmatrix} h \\ hu \\ hv \end{bmatrix}$$

Luego, tenemos los dos vectores de flujo:

$$F(U) = \begin{bmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \\ huv \end{bmatrix}, \qquad G(U) = \begin{bmatrix} hv \\ hvu \\ hv^2 + \frac{1}{2}gh^2 \end{bmatrix},$$

Y, finalmente, tenemos el vector de origen:

$$S(U) = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix}$$

La unión de estos 4 términos, forma la ecuación vectorial que hemos mencionado previamente:

$$U_t + F(U)_x + G(U)_y = S(U) \tag{3.38}$$

Hay que tener en cuenta, que en el vector $S(U)$ podría haber más términos como las fuerzas de Coriolis, las fuerzas del viento, o la fricción del fondo, pero en nuestro caso las hemos desestimado para poder seguir con la aproximación numérica de las ecuaciones diferenciales parciales, asumiendo en nuestro caso que $S(U) = 0$. Este razonamiento, junto con que estamos tratando las ecuaciones de shallow water en una sola dimensión, nos permite simplificar la ecuación (3.38):

$$U_t + F(U)_x = 0 \tag{3.39}$$

Donde la refactorización de los dos términos restantes, se definen para una dimensión de la siguiente forma:

$$U = \begin{bmatrix} h \\ hu \end{bmatrix}, \qquad F(U) = \begin{bmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \end{bmatrix}$$

La ecuación (3.39) que representa la ley de conservación, puede ser escrita en forma integral:

$$\oint (U dx - F(U) dt) = 0 \tag{3.40}$$

Finalmente, hay que tener en cuenta que la ecuación (3.40) puede admitir soluciones discontinuas como agujeros, a diferencia de la ecuación (3.39)

3.2.5. Ecuaciones de Shallow Water en 1D. Ecuaciones de Saint-Venant.

En este apartado veremos las ecuaciones de Shallow Water en 1D, que son las que posteriormente ejemplificaremos. Dichas ecuaciones también son conocidas como las ecuaciones de Saint-Venant, que es como las nombraremos en este apartado.

Las ecuaciones de Saint-Venant hacen referencia al matemático francés Adhémar Jean Claude Barré de Saint-Venant, el cual desarrolló estas ecuaciones en 1871. Estas, son un conjunto de ecuaciones diferenciales cuyo propósito es el de la modelización de los cambios de caudal y nivel de un líquido a lo largo de un espacio unidimensional, que en nuestro caso, usaremos como ejemplo el caudal y nivel del agua de un rio.

Las ecuaciones de Saint-Venant, son un caso particular de las ecuaciones de Shallow Water, derivado del sistema de Euler en tres dimensiones anteriormente descrito y suponiendo cosas como una pequeña cantidad de fluido, presión hidrostática, etc. Todo esto nos permite hacer una descripción del flujo en un determinado punto del dominio $(x, y) \in \mathbf{R}^2$ cuando evoluciona en un tiempo $t \geq 0$.

Además, como el flujo es unidimensional, se supone también que las variaciones de velocidad en vertical y en horizontal son pequeñas, con lo que no existen cambios de dirección bruscos dentro del canal.

Partimos de dos ecuaciones de suma importancia para la modelización del flujo del rio, las

cuales derivan de las anteriores ecuaciones de Shallow Water (3.39 - 3.40) pero en 1D:

$$A_t + (Au)_x = 0 \tag{3.41}$$

y

$$(Au)_t + (Au^2)_x + gAh_x = gA(b_x - b_f) \tag{3.42}$$

Donde A es el área de la sección transversal, que obtenemos integrando dicha sección transversal $r(x, z)$, entre la altura del caudal del río y el fondo, $h(x)$ y 0.

$$A = \int_0^{h(x)} r(x, z)\, d_z$$

Además, tenemos la anchura de la superficie $r(x, h) = B(x)$, la pendiente geométrica b_x, la pendiente de fricción o pendiente motriz b_f y la tasa de flujo volumétrico $Q = Au$, que es la cantidad de volumen de fluido que pasa a través de nuestra sección transversal (de nuestra área), en un tiempo dado. Podemos observar todas estas definiciones de forma gráfica en la Figura 30.

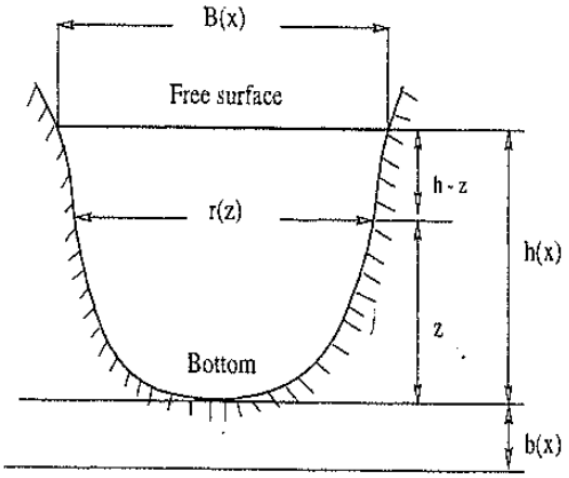


Fig. 30: Sección transversal de un río.
Shock-Capturing Methods for Free-Surface Shallow Flows, Eleuterio F. Toro

Partimos de las ecuaciones en forma conservativa, y para representarlas en forma primitiva, vamos a tener que desarrollarlas. De modo que en primer lugar expandiremos la ecuación (3.41):

$$A_t + uA_x + Au_x = 0 \tag{3.43}$$

Haremos lo mismo con (3.42):

$$(Au)_t + (Au^2)_x + gAh_x = gA(b_x - b_f) =$$

$$uA_t + Au_t + u^2A_x + 2Auu_x + gAh_x = gA(b_x - b_f)$$

Despejamos A_t de la ecuación previamente expandida (3.43):

$$A_t = -uA_x - Au_x$$

Sustituimos la igualdad en la ecuación que estamos expandiendo y terminamos de operar:

$$u(-uA_x - Au_x) + Au_t + u^2A_x + 2Auu_x + gAh_x = gA(b_x - b_f) =$$

$$-u^2A_x - Auu_x + Au_t + u^2A_x + 2Auu_x + gAh_x = gA(b_x - b_f) =$$

$$Au_t + Auu_x + gAh_x = gA(b_x - b_f) =$$

$$A(u_t + uu_x + gh_x) = gA(b_x - b_f) =$$

$$u_t + uu_x + gh_x = g(b_x - b_f)$$

Finalmente, teniendo en cuenta que $B = \frac{\partial A}{\partial h}$ como vemos en [1], entonces, siendo h la altura y B una constante de ancho de banda del canal, tenemos:

$$B = \frac{\partial A}{\partial h} = \frac{\partial A}{\partial x} \cdot \frac{\partial x}{\partial h} = \frac{\partial A_x}{\partial h_x}$$

Despejando h_x:

$$h_x = \frac{A_x}{B}$$

Por lo que, sustituyendo esta última igualdad en nuestra ecuación restante, obtenemos finalmente:

$$u_t + uu_x + g\frac{A_x}{B} = g(b_x - b_f) \tag{3.44}$$

El sistema de equaciones obtenido formado por (3.43) y (3.44), podemos escribirlo de forma cuasilineal:

$$W_t + AW_x = S$$

Donde los términos vienen dados por:

$$W = \begin{bmatrix} A \\ u \end{bmatrix}, \; A = \begin{bmatrix} u & A \\ g/B & u \end{bmatrix}, \; S = \begin{bmatrix} 0 \\ gA(b_x - b_f) \end{bmatrix}.$$

3.3. Ejemplo Numérico: La rotura de presa

Finalmente, para dar uso a las ecuaciones obtenidas anterioremente a modo de ejemplo, plantearemos el problema de la rotura de presa. Antes de ponernos con el problema, describiremos brevemente dos apartados necesarios para su resolución.

3.3.1. Problema de Riemann

Un problema de Riemann, se describe como un problema con una función a trozos con una sola discontinuidad, que viene dada a partir de una ley de conservación y tiene una condición inicial dada por dicha función. El problema de Riemann es una generalización del problema de la rotura de presa con el que vamos a trabajar.

Podemos representar un simple problema de Riemann por la siguiente ecuación de avección:

$$\begin{cases} u_t + cu_x = 0, \quad x \in \mathbf{R}, t \geq 0 \\ \\ u(x,0) = \begin{cases} u_L, & x < 0, \\ u_R, & x > 0. \end{cases} \end{cases}$$

Aqui vemos el problema representado de forma gráfica.

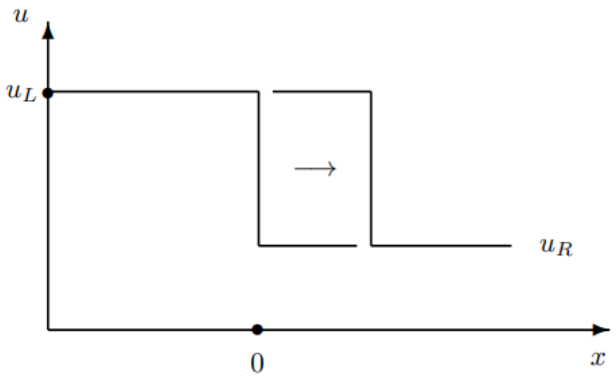


Fig. 31: Problema de Riemann.

Como vemos en la anterior Figura, la diferencia de altura $|u_L - u_R|$, viaja de izquierda a derecha, con velocidad c ($c > 0$). Dicha discontinuidad se propaga a lo largo del plano OX,

empezando desde el punto $x = 0$, y con tiempo $t = 0$, cuya ecuación viene dada por $x = ct$. Entonces, la solución al problema de Riemann viene dada por la siguiente función:

$$u(x,t) = \begin{cases} u_L, & x - ct < 0 \\ u_R, & x - ct > 0 \end{cases}$$

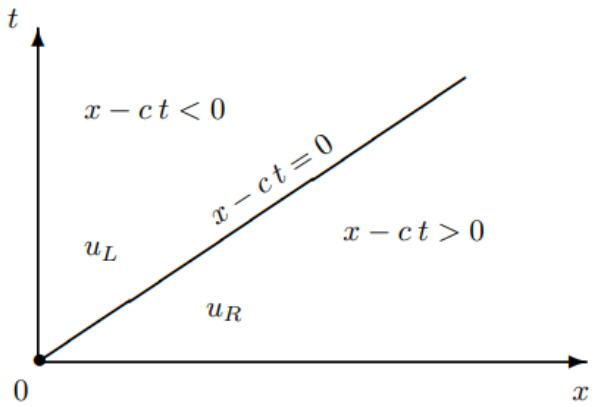


Fig. 32: Solución al problema de Riemann.

3.3.2. Lax-Friedrichs

La resolución de este ejemplo se realiza mediante un método de *diferencias finitas*, ya que, en general, no es posible encontrar soluciones exactas a estas ecuaciones. Dicho método, nos permite calcular las soluciones de forma numérica, con lo que nos permite obtener una aproximación al comportamiento de dichas ecuaciones diferenciales en derivadas parciales.

Los métodos de diferencias finitas, tratan de subdividir el plano (x, t) en volúmenes finitos de tal forma que un determinado punto del plano subdividido se define como $I_i = [x_{j-\frac{1}{2}}, x_{j+\frac{1}{2}}]$, cuyo tamaño del volumen finito viene dado por el cociente de la longitud del plano y entre el número de puntos en el que queramos dividirlo. Y de cada una de estas celdas, sacamos una aproximación numérica que resolveremos y de la cual obtendremos esa solución aproximada.

El método numérico que usaremos para la resolución de este ejemplo es el de *Lax-Friedrichs*. El método de *Lax-Friedrichs* es un método numérico que permite la resolución de ecuaciones hiperbólicas en derivadas parciales basado en las diferencias finitas, y cuya fórmula para

la resolución de estas ecuaciones es la siguiente:

$$f^n_{j+\frac{1}{2}} = \frac{1}{2}(f(u_j) + f(u_{j+1})) - \frac{1}{2\lambda}(u_{j+1} - u_j) \tag{3.45}$$

Donde u se trata de una función de dos variables independientes $x \in (a, b)$ y $t \geq 0$. Tal y como hemos mencionado en la explicación del método de diferencias finitas, subdividimos el plano (x, t) en un mallado con longitudes $\triangle x = h$, y $\triangle t = k$, $\lambda = k/h$. Además, vamos a definir cada uno de estos puntos como $x_j = a + jh$ donde $j = 0, 1, 2, ...N$, y $t_n = nk$, con $n = 0, 1, 2...N$. Definimos entonces una aproximación numérica de la función u en un determinado punto del mallado (x_j, t_n), como $u^n_j = u(x_j, t_n)$.

Donde también tenemos la función flujo numérico:

$$f^n_{j+\frac{1}{2}} \sim \frac{1}{k} \int_{t_n}^{t_{n+1}} f(u(x_{j+\frac{1}{2}}, t))dt$$

Y la solución se aproxima a medida que evoluciona en el tiempo:

$$u^{n+1}_j = u^n_j - \lambda \left[f^n_{j+\frac{1}{2}} - f^n_{j-\frac{1}{2}} \right]$$

3.3.3. La rotura de presa

Empezamos considerando un canal de agua, cuya longitud es de 20m. En la mitad del canal se encuentra una pared con altura de 10m. A la izquierda de esta pared, encontramos 1m de agua de altura, es decir, la misma que la de la pared, y a la derecha, una profundidad de 0.25m de agua. Hemos de considerar que inicialmente, el agua está en calma, por lo que la velocidad inicial es de 0m/s. Vemos este ejemplo ilustrado en la siguiente Figura.

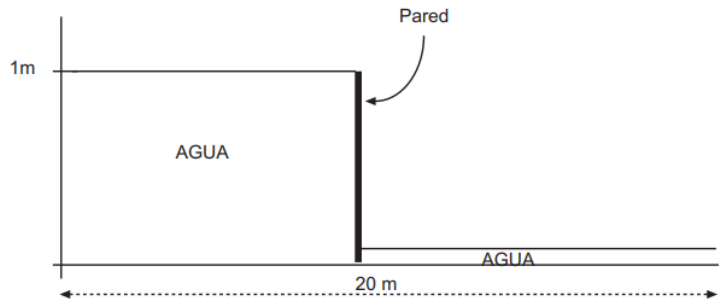


Fig. 33: Planteamiento gráfico del problema de la rotura de presa.

Ahora, supongamos que en cierto momento $t = 0$, la pared se rompe. Este problema, podemos modelizarlo con las ecuaciones de Shallow Water en 1D o Saint-Venant. Por lo tanto, utilizando la ecuación (3.36), definimos la condición inicial:

$$W(x,0) = \begin{cases} \begin{pmatrix} 1 \\ 0 \end{pmatrix}, & x < 10, \\ \begin{pmatrix} 0{,}25 \\ 0 \end{pmatrix}, & x > 10, \end{cases}$$

A partir de esta condición inicial, calcularemos una solución numérica para varios valores t y así ver su comportamiento. Además, subdividiremos la malla en 200 puntos y usaremos una condición de estabilidad o número de Courant, CFL=0.8. Para los métodos de primer orden, satisface $0 < CFL \leq 1$. El número de Courant se obtiene mediante el intervalo de tiempo $\triangle t$, intervalo de espacio $\triangle x$, y la velocidad de la siguiente forma:

$$C = \frac{v \cdot \triangle t}{\triangle x}$$

La condición de Courant, se usa en problemas de ecuaciones diferenciales en derivadas parciales, con ciertas condiciones, ya que, si el incremento del tiempo es demasiado grande, se producen inestabilidades en la simulación del programa (Fig. 34), con lo que este incremento de tiempo solo podría aumentar si aumentara también el incremento del espacio x.

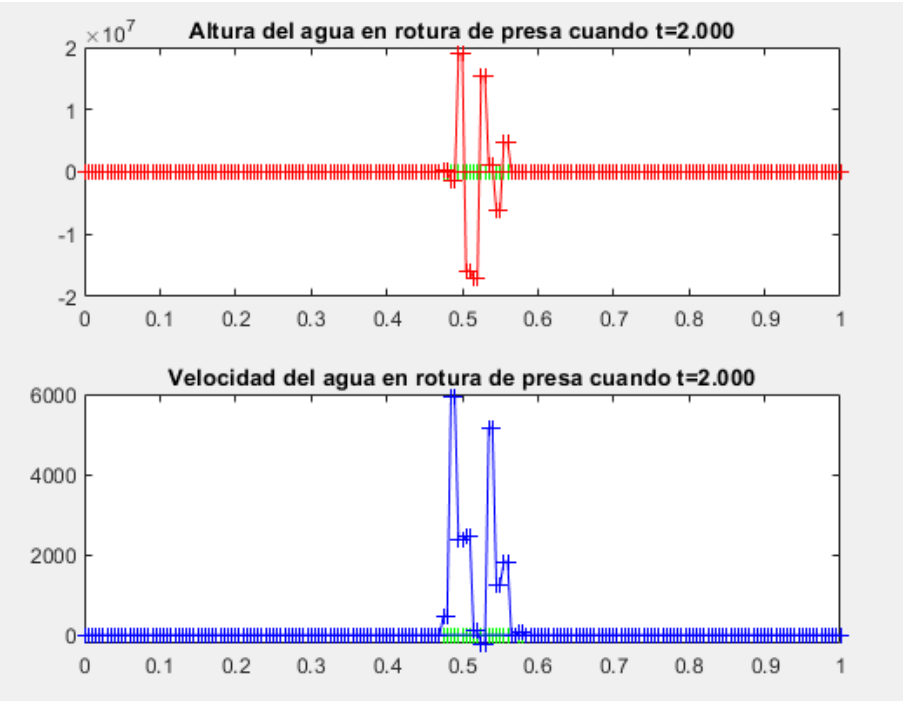


Fig. 34: Inestabilidad en la simulación de la rotura de presa.

Tal y como vemos en la Figura, al aumentar el incremento de 0.001 a 0.002, como no aumenta el incremento del espacio, se producen inestabilidades, donde vemos que el altura llega a superar los 20 millones de metros y la velocidad alcanza los 6 km/s. Usaremos 3 funciones para la resolución, dos de ellas auxiliares a la principal. La primera función *exact high*, calcula y dibuja la solución exacta de nuestro problema.

En la función, vemos como inicialmente dividimos x en una malla de 200 puntos ($x = 0 : 0{,}005 : 1; => 1/0{,}005 = 200$), tal y como hemos dicho en el planteamiento del problema. Además, hacemos lo mismo con la variable t que representa el tiempo, donde vemos que *tstop* es la variable de tiempo que indicará el tiempo durante el cual se forma el movimiento del agua tras romper la pared.

Después de calcular las condiciones iniciales, dividimos en 4 condiciones la altura a la que estará el agua, la cual depende de la variable tiempo, a mayor tiempo, la altura del agua a la izquierda de la pared disminuirá e irá equilibrándose. Estas 4 condiciones vienen dadas por la solución exacta al problema de Riemann resuelto por George Gabriel Stokes para el caso del fondo siempre positivo, es decir, el fondo siempre está mojado, tal y como nuestro problema, el canal nunca llega a estar seco.

Estas 4 condiciones vienen dadas por cuatro estados constantes de las olas formadas al

romper la pared. Dadas las condiciones iniciales se tiene que determinar por estas 4 condiciones, el tipo de ola que se forma, ya sean olas de choque (compresión) o de rarefacción[1] (expansión).

RoturaDePresa/exact_high_pA.m

```
function exact_high_pA(tstop)
%Repartimos la longitud x en una malla de 200 puntos
x=0:0.005:1; I=length(x);
t=0:0.001:tstop; N=length(t);

%Condiciones iniciales
s=2.957918120187525;
g=9.8;
c_2=sqrt(g/4*(sqrt(1+16*s*s/g)-1));
u_2=s-g/8/s*(1+sqrt(1+16*s*s/g));
p_0=0.5;

%Matriz de NxI, la cual representa la altura calculada para cada
   determinado punto x, y tiempo t
h=zeros(N,I);
for n=1:N
    for i=1:I
        if x(i)<1/2-t(n)*sqrt(g)
            h(n,i)=1;

        elseif (x(i)>=1/2-t(n)*sqrt(g))&&(x(i)<=(u_2-c_2)*t(n)+1/2)
            h(n,i)=1/9/g*(2*sqrt(g)-(2*x(i)-1)/2/t(n))*(2*sqrt(g)
                -(2*x(i)-1)/2/t(n));

        elseif (x(i)>(u_2-c_2)*t(n)+1/2)&&(x(i)<=s*t(n)+1/2)
            h(n,i)=1/4*(sqrt(1+16*s*s/g)-1);

        elseif x(i)>s*t(n)+1/2
            h(n,i)=p_0;
        end
    end
end
plot(x,h(N,:),'r')
end
```

[1]La rarefacción es un proceso por el cual un cuerpo disminuye su densidad.

El resultado gráfico es el siguiente, donde la línea verde vertical representa la pared y la línea roja representa la altura que tiene que tener el agua en un determinado tiempo, tras romperse la pared.

Fig. 35: Solución teórica - Altura del agua en rotura de presa, t=0.05s.

Vamos a ver que esta forma escalonada del agua se pierde al cabo de un tiempo, ya que, si aumentamos el tiempo a 5 segundos, el agua termina por estabilizarse, con lo que tiene la misma altura durante todo el plano OX.

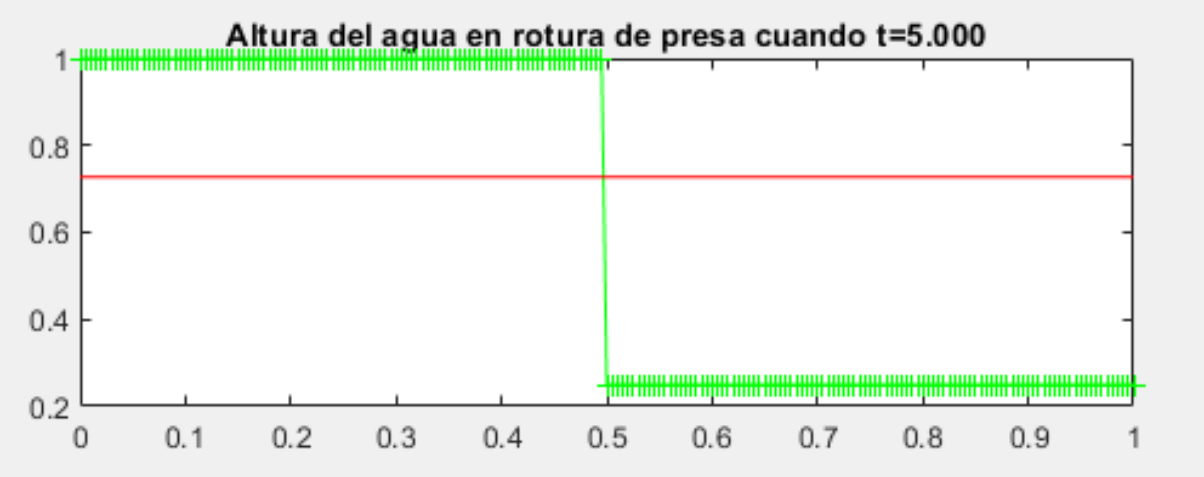


Fig. 36: Solución teórica - Altura del agua en rotura de presa, t=5s.

A continuación, la segunda función que vamos a ver, representa la velocidad del agua en cada uno de estos 200 puntos mientras transcurre el tiempo.

RoturaDePresa/exact_velocity_pA.m

```
function exact_velocity_pA(tstop)
%Repartimos la longitud x en una malla de 200 puntos
x=0:0.005:1; I=length(x);
t=0:0.001:tstop; N=length(t);

%Condiciones iniciales
s=2.957918120187525;
g=9.8;
c_2=sqrt(g/4*(sqrt(1+16*s*s/g)-1));
u_2=s-g/8/s*(1+sqrt(1+16*s*s/g));

%Matriz de NxI, la cual representa la velocidad calculada para cada
    determinado punto x, y tiempo t
u=zeros(N,I);
for n=1:N
    for i=1:I
        if x(i)<1/2-t(n)*sqrt(g)
            u(n,i)=0;
        elseif (x(i)>=1/2-t(n)*sqrt(g))&&(x(i)<=(u_2-c_2)*t(n)+1/2)
            u(n,i)=1/3/t(n)*(2*x(i)-1+2*t(n)*sqrt(g));
        elseif (x(i)>(u_2-c_2)*t(n)+1/2)&&(x(i)<=s*t(n)+1/2)
            u(n,i)=u_2;
        elseif x(i)>s*t(n)+1/2
            u(n,i)=0;
        end
    end
end
plot(x,u(N,:),'b')
end
```

Como vemos en las siguientes dos gráficas, a mayor valor del tiempo, la velocidad del agua también empieza a aumentar porque se consigue movilizar el agua del canal entero.

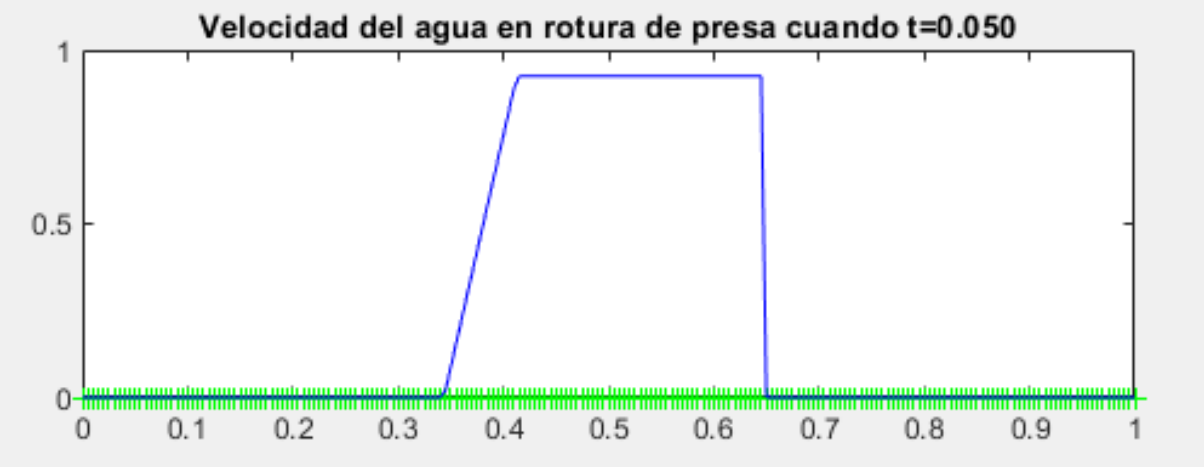


Fig. 37: Solución teórica - Velocidad del agua en rotura de presa, t=0.05s.

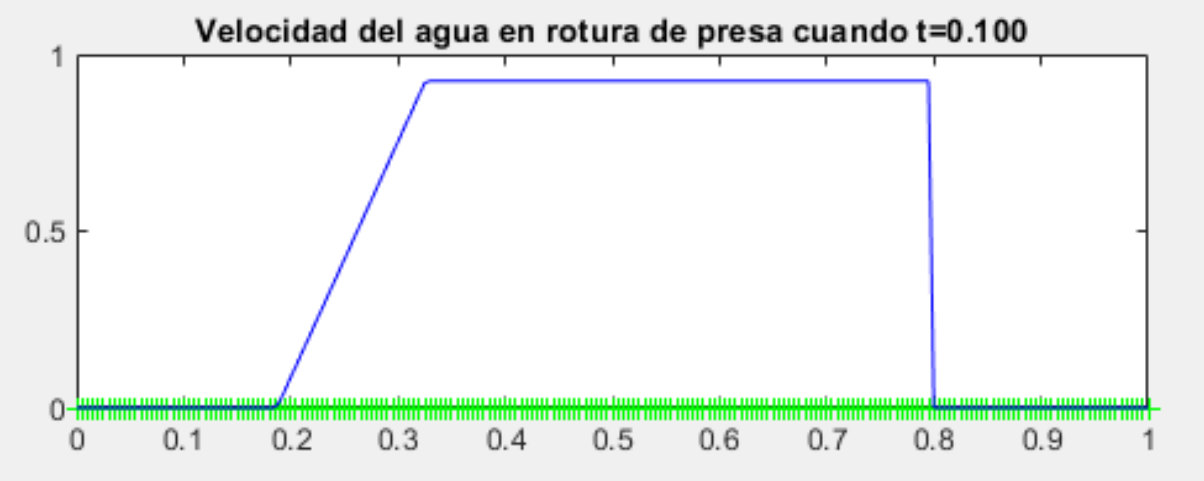


Fig. 38: Solución teórica - Velocidad del agua en rotura de presa, t=0.1s.

Finalmente, con la última función aplicaremos el método de *Lax-Friedrichs*, con el que conseguiremos una aproximación de la solución numérica. En primer lugar, inicializamos las variables fijas como la gravedad, la altura del agua en la parte derecha de la pared, etc. También inicializamos la malla de 200 puntos y de la misma forma para el tiempo, a mayor valor de *tstop*, más tiempo estará la función ejecutándose.

Además, hemos de inicializar las matrices que contendrán los valores numéricos, y ponemos ya las condiciones iniciales del problema planteado.

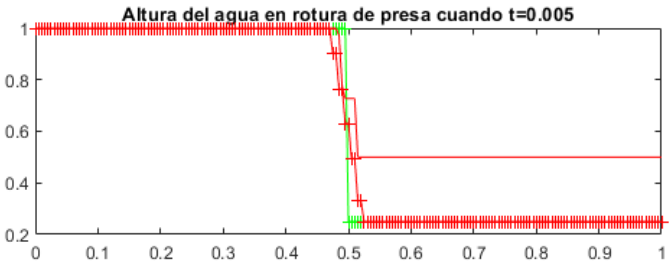
Luego, al entrar en el bucle, se van rellenando los valores con el método de *Lax-Friedrichs*, a la par que se van dibujando estas soluciones numéricas para una altura h y para un tiempo t.

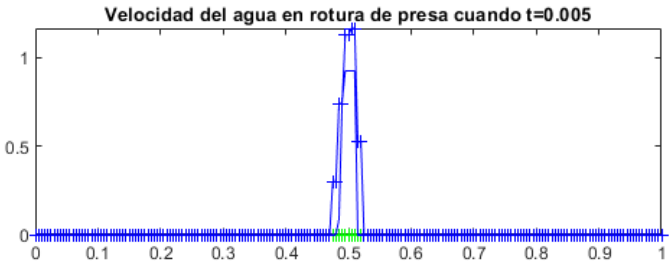
RoturaDePresa/Dam_1D.m

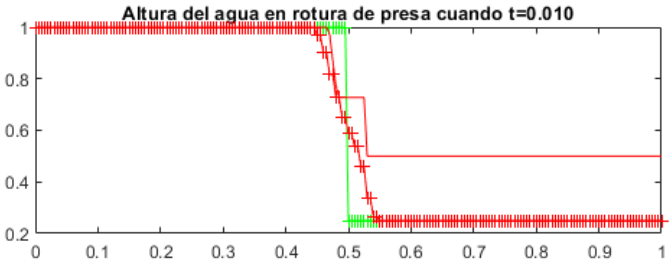
```
function Lax_Friedrichs_prA
clear all
close all
clc
g=9.8; % Gravedad
p_0=0.25; % Altura del agua a la derecha de la pared
tstop=0.5; % Tiempo final

% Vectores de tiempo y de movimiento en x, I = 1/0.005 = 200 =>
    malla de 200 puntos
dt=.002; t=(0:dt:tstop); N=length(t);
dx=.005; x=(0:dx:1); I=length(x);

% Inicializamos matrices de NxI
u=zeros(N,I); Q1=zeros(N,I);
h=zeros(N,I); Q2=zeros(N,I);

% Condiciones iniciales de nuestro problema
u(1,:)=0;
h(1,1:(I-1)/2)=0.1; h(1,(I+1)/2:end)=p_0;
Q1(1,:)=h(1,:);
Q2(1,:)=u(1,:).*Q1(1,:);

figure(1)
for n=2:N
    Q1(n,:)=h(n,:);
    Q2(n,:)=Q1(n,:).*u(n,:);
    for i=2:I-1

        Q1(n,i)=(h(n-1,i+1)+h(n-1,i-1))/2-dt/dx*(h(n-1,i+1)*u(n-1,i
```

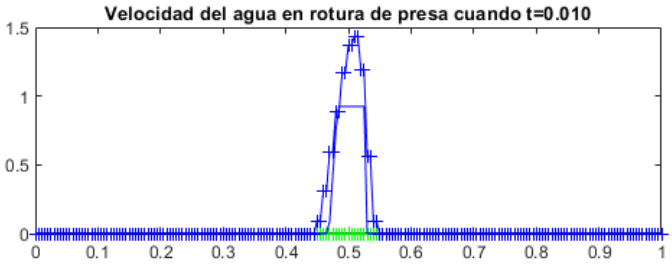
```
            +1)-h(n-1,i-1)*u(n-1,i-1))/2;
        Q2(n,i)=(h(n-1,i+1)*u(n-1,i+1)+h(n-1,i-1)*u(n-1,i-1))/2-dt/
            dx*(DFX(h(n-1,i+1),u(n-1,i+1),g)-DFX(h(n-1,i-1),u(n-1,i
            -1),g))/2;
        h(n,i)=Q1(n,i);
        u(n,i)=Q2(n,i)/Q1(n,i);
    end

    % Condiciones de frontera de la pared
    u(n,1)=u(n,2); u(n,I)=u(n,I-1);
    u(n,1)=0; u(n,I)=0;
    h(n,1)=h(n,2); h(n,I)=h(n,I-1);

    % Grafico de la altura
    subplot(2,1,1)
    plot(x,h(1,:),'g+-')
    hold on
    % Funcion exacta de la altura
    exact_high_pA(tstop)
    plot(x,h(n,:),'r+-')
    title(sprintf('Altura del agua en rotura de presa cuando t=%4.3f
        ',tstop))
    hold off

    % Grafico de la velocidad
    subplot(2,1,2)
    plot(x,u(1,:),'g+-')
    hold on
    % Funcion exacta de la velocidad
    exact_velocity_pA(tstop)
    plot(x,u(n,:),'b+-')
    title(sprintf('Velocidad del agua en rotura de presa cuando t
        =%4.3f',tstop))
    hold off
    pause(0.0001)

end

function F=DFX(a,b,c)
F=a*b*b+1/2*c*a*a;  % h*u^2+(1/2)*g*h^2
end
end
```

En los siguientes resultados gráficos, iremos viendo la altura del agua a medida que vamos viendo también la velocidad de forma simultánea. En las primeras dos imágenes, vemos como la altura del agua está a 1m a la izquierda y a 0.25m a la derecha, y una vez se rompe la pared, el agua empieza a llenarse por la derecha tal y como vemos en la función aproximada (la que se divide en 200 puntos). A su vez, vemos como la velocidad del agua llega casi ya a los 1.5m/s en el centro donde se está produciendo la movilización, mientras que, a los lados, la velocidad del agua sigue intacta puesto que la onda aún no ha llegado a esos puntos.

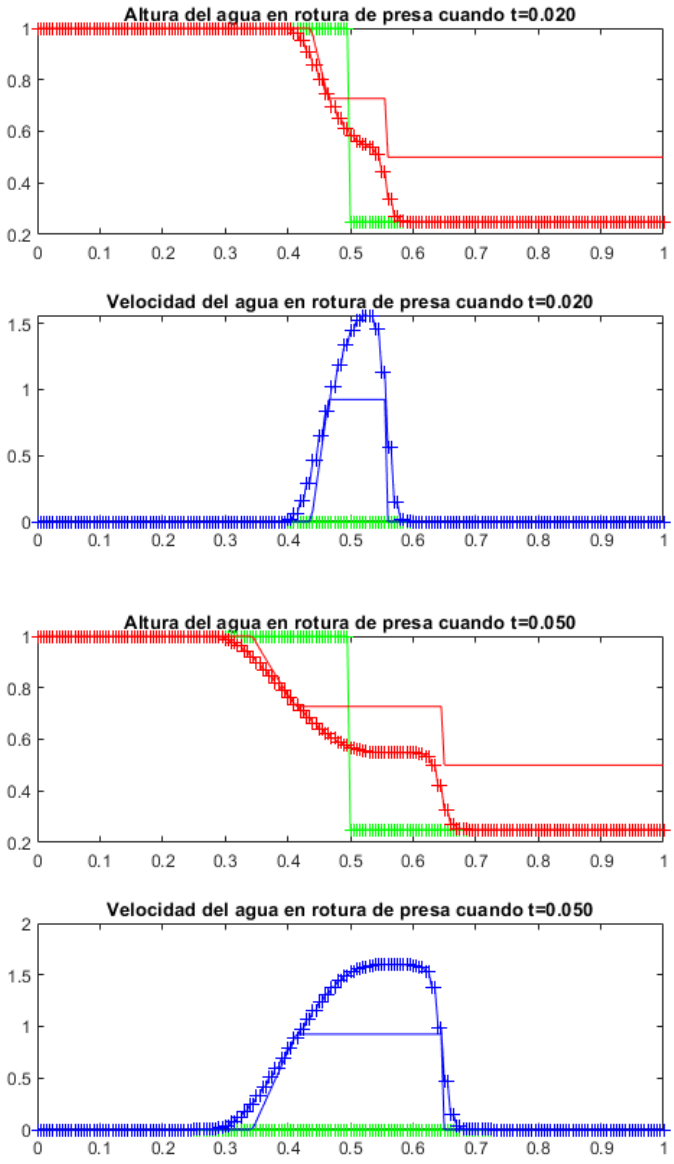




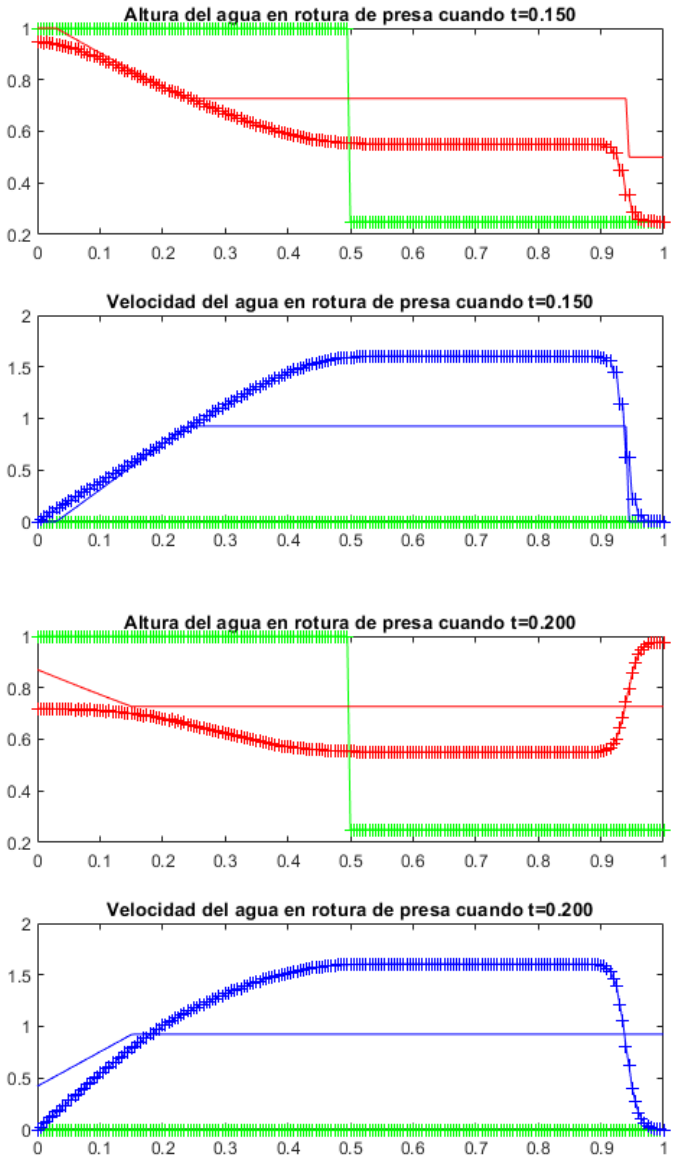




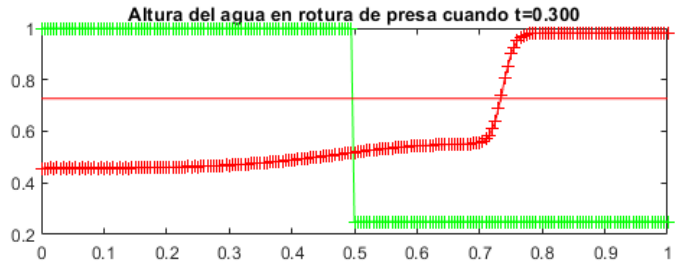
Al llegar a t=0.02, el agua sigue cayendo y la función aproximada se asemeja más al movimiento real del agua que la propia función exacta, vemos además, que la velocidad aumenta de forma más agresiva por la derecha, ya que, si nos fijamos, la separación entre los puntos en el movimiento que se está formando a la derecha es mucho mayor que la que se está formando a la izquierda, que de hecho sigue una forma más parabólica y menos vertical.

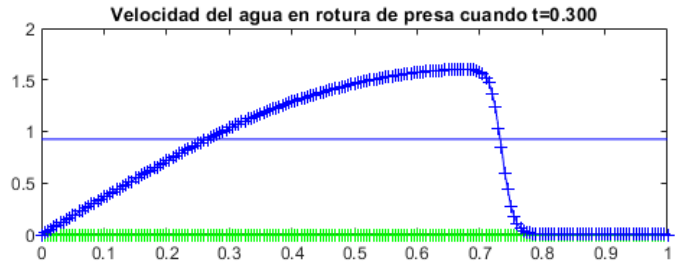


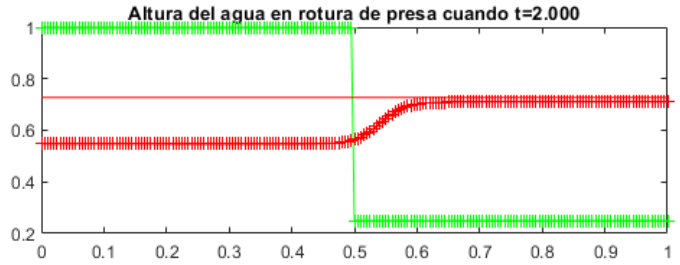
Entre t=0.15 y t=0.2, la ola está a punto de llegar al límite del canal, vemos que prácticamente toda el agua a excepción de unos pocos puntos, la velocidad es $v \geq 0$ mientras el agua sigue cayendo. Cuando finalmente el agua que cae consigue llegar al límite del canal, este rebota (hasta el límite de 1m de altura) y el movimiento sigue esta vez de derecha a izquierda, además, vemos que tras alcanzar el límite, la velocidad vuelve a bajar hasta 0 en el extremo derecho.

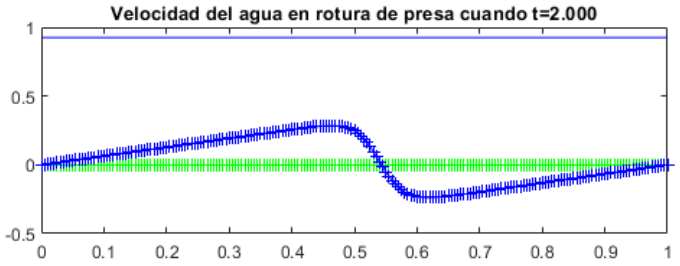


Finalmente, el agua sigue su curso y entre el intervalo de t=0.3 a t=2, vemos que el agua se estabiliza poco a poco y a la larga termina acercándose a la función exacta, la cual está estabilizada entre una altura de 0.6 y 0.8.









Capítulo 4

Conclusiones

Llegados a este punto y tras la finalización tanto del curso como de las prácticas, debo decir que estoy muy contento y satisfecho por lo realizado durante todo el año. En primer lugar, estoy satisfecho con las prácticas por todo lo aprendido y por el trato recibido. Realizar las prácticas ha supuesto un antes y un después en lo que se refiere a conocimientos informáticos y experiencia laboral, he aprendido de forma más rápida, mejor y con mayor rendimiento. He aprendido varios lenguajes de programación durante mi estancia, los cuales pienso que son importantes y, sobretodo, interesantes, con los que podré seguir dedicándome al ámbito del desarrollo de aplicaciones de aquí en adelante. Además, he aprendido mucho sobre el trabajo en equipo y sobre nuevas formas de trabajo, que son más dinámicas a la hora de programar. También quiero recalcar que, aunque en los estudios de grado no he dado ninguno de los 3 lenguajes de programación que aprendí durante mi estancia, la preparación de esta y la buena base con la que nos preparan, me ha permitido aprender mucho más rápido estos lenguajes y aplicar muchas de las cosas aprendidas durante este periodo.

En segundo lugar, estoy también bastante satisfecho con el trabajo de fin de grado realizado, en donde me gustaría matizar, antes que nada, que mi tutor, Vicente, ha sido de gran ayuda durante todo el curso, explicándome las partes que no entendía, resolviendo dudas puntuales a todas horas y sobretodo tenido mucha paciencia y dedicación. Este trabajo me ha permitido adentrarme un poco en la parte más teórica de las matemáticas y de las deducciones sobre varios modelos que, tras pasar tiempo investigando y buscando entre libros y artículos, para contrastar y encontrar información, empiezas a sentirte más curioso e identificado con el tema, lo que te permite seguir buscando y buscando más información para ver cuál es el alcance real de un proyecto como este.

Esto me da paso para ver y describir los siguientes posibles desarrollos de este proyecto. El desarrollo podría tomar varios rumbos, por ejemplo, durante el desarrollo hemos desestimado

varios términos que afectarían y cambiarían nuestro desarrollo, como lo son las fuerzas de Coriolis, las fuerzas del viento, o la fricción del fondo. Por otra parte, tal y como indica el título, el estudio se basa en las ecuaciones que rigen el movimiento en una sola dimensión (aunque hayamos usado las 2 o 3 dimensiones en varias ocasiones para permitirnos un estudio más general), con lo que se podría seguir el trabajo perfectamente, pero enfocándolo hacia más de una dimensión. Además, se han extraído y obviado algunas fórmulas para el ejemplo gráfico de la rotura de presa, donde se puede profundizar mucho más en los problemas de Riemann.

Bibliografía

[1] Eleuterio F. Toro, *Shock-Capturing Methods for Free-Surface Shallow Flows*. Manchester Metropolitan University, UK, 2001.

[2] Eleuterio F. Toro, *Rienmann Solvers and Numerical Methods for Fluid Dynamics*. Manchester Metropolitan University, UK, 1999.

[3] Raquel Taboada Vázquez, *Modelos de aguas poco profundas obtenidos mediante la técnica de desarrollos asintóticos*. Universidade da Coruña, A Coruña, 2006.

[4] Stoker, J.J., *Water waves: The Mathematical Theory with Applications*. Dover Publications, Inc. Mineola, New York, 2019. Originally published: New York: Interscience Publishers, 1957.

[5] Emmanuel Audusse, Marie-Odile Bristeau, Benoît Perthame. *Kinetic Schemes for Saint-Venant Equations with Source Terms on Unstructured Grids*. [Research Report] RR-3989, INRIA. 2000. ffinria00072657

[6] Vicente Martínez. *Métodos Matemáticos en Ecuaciones en Derivadas Parciales*. Universitat Jaume I, 2008.